**AFRL-RI-RS-TR-2009-157**
**In-House Final Technical Report**
**June 2009**


# INVESTIGATING ARCHITECTURAL ISSUES IN NEUROMORPHIC COMPUTING

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.***

**STINFO COPY**



**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-157 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
DUANE A. GILMOUR, Chief
Computing Tech Applications Branch

/s/
EDWARD J. JONES, Deputy Chief
Advanced Computing Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| JUNE 2009 | Final | Jan 06 – Sep 08 |

**4. TITLE AND SUBTITLE**

INVESTIGATING ARCHITECTURAL ISSUES IN NEUROMORPHIC COMPUTING

**5a. CONTRACT NUMBER**
In-House 231TINHP

**5b. GRANT NUMBER**
N/A

**5c. PROGRAM ELEMENT NUMBER**
61102F

**6. AUTHOR(S)**

Richard W. Linderman, Daniel Burns, Michael Moore, Qing Wu, Qinru Qiu, and Tarek Taha

**5d. PROJECT NUMBER**
231T

**5e. TASK NUMBER**
IN

**5f. WORK UNIT NUMBER**
HP

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

AFRL/RITB
525 Brooks Rd.
Rome NY 13441-4505

**8. PERFORMING ORGANIZATION REPORT NUMBER**
N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFRL/RITB
525 Brooks Rd.
Rome NY 13441-4505

**10. SPONSOR/MONITOR'S ACRONYM(S)**
N/A

**11. SPONSORING/MONITORING AGENCY REPORT NUMBER**
AFRL-RI-RS-TR-2009-157

**12. DISTRIBUTION AVAILABILITY STATEMENT**
*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-1883*

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
This effort has explored the issues associated with the efficient mapping of neuromorphic computing strategies onto advanced computational architectures. This multidisciplinary effort combined concepts and research from diverse fields including computer architecture, neuroscience, cognitive psychology, cognitive modeling, dynamical systems, software and computer engineering. It explored multiple columnar cortical models reported in the literature, and produced new models by combining ideas with insights developed by the research team. These models range in scale of abstraction from cell assemblies of individual minicolumns to models that represent abstractions of hundreds of thousands of synapses and neurons. Selected models were also emulated. Columnar model software produced by this effort includes C code and FPGA VHDL code. The VHDL code consists of "accelerations" of Bayesian tree recall algorithms, and Brain State in the Box point attractors. The C code consists of these algorithms, models of minicolumns and functional columns, spiky neuron models, and Confabulation models.

**15. SUBJECT TERMS**
Neuromorphic Computing, advanced computational architectures, Columnar Models, Large Scale Cortical Models, Columnar software models

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Stanley Lis |
| U | U | U | UU | 96 | 19b. TELEPHONE NUMBER (*Include area code*) N/A |

The following people contributed to this work:

| **Individual** | **Involvement** |
|---|---|
| Dr. Richard Linderman, AFRL/RI | PI, Direction, supervision, efficient model representations, hybrid models, and mappings onto Cell BE Cluster |
| Mr. Daniel Burns, AFRL/RITC | Confabulation, developed full rank vector confabulator, and worked closely with Wu & Qui on hybrid study, metrics. |
| Dr. Thomas Renz, AFRL/RITC | Advisement on literature review |
| Mr. Michael Moore, ITT Industries | Multidisciplinary literature review, sparse data confabulator, Spiking neuron investigation, BSB efficacy study, Pub/Sub inclusion, BSB Hybrid V1 model development |
| Mr. Dennis Fitzgerald, ITT Industries | Advisement on architectural concepts Review |
| Dr. Qing Wu, SUNY Binghamton, Dept of Electrical and Computer Engineering | BSB FPGA & CELL-BE investigation, BSB-Confabulation hybrid |
| Dr. Qinru Qiu SUNY Binghamton, Dept of Electrical and Computer Engineering | Confabulation investigations: Hashing, MPI-parallelism & FPGA acceleration, BSB-Confabulation hybrid |
| Dr. Tarek Taha, Clemson University, Department of Electrical and Computer Engineering | Hierarchical Bayesian model investigations: efficacy, FPGA and MPI-parallelism accelerations |
| Ken Rice, Clemson, Department of Electrical and Computer Engineering | Hierarchical Bayesian model investigations: efficacy, FPGA and MPI-parallelism accelerations |
| Chris Vutsinas, Clemson, Department of Electrical and Computer Engineering | Hierarchical Bayesian model investigations: efficacy, FPGA and MPI-parallelism accelerations |
| Andrew C. Flack, University of Rochester | Confabulation recall metrics & testing |

## TABLE OF CONTENTS

## LIST OF FIGURES

**LIST OF TABLES**

# 1. Summary

This effort has explored the issues associated with the efficient mapping of neuromorphic computing strategies onto advanced computational architectures. The computing performed by neurological systems produces cognitive phenomena that have been high value, yet elusive, goals of computational researchers. Neuromorphic computing, as evident in primate brains, uses massive collections of modest speed synapses and neurons operating asynchronously in parallel. This computation is characteristically:

- Performed with precision and robustness;
- Accomplished with very low power consumption;
- Performed in real time, allowing the fusion of sensing, planning, and interaction with the environment
- Performed without programming, based on experience

A characteristic challenge of the effort is its multidisciplinary nature. It combined ideas and research from diverse fields including computer architecture, neuroscience, cognitive psychology, cognitive modeling, dynamical systems, belief systems, software and computer engineering.

The effort has explored multiple columnar cortical models reported in the literature, and produced new models by combining ideas with insights developed by the team. These models range in scale of abstraction from cell assemblies of individual minicolumns to models that represent abstractions of hundreds of thousands of synapses and neurons. In each case, effort was made to understand neuron-based computational underpinnings, the cognitive efficacy of the model, the fit of the digital emulation of the model to computer architectural features, and the scaling of the model into a full-scale system. Selected models were also emulated.

The results suggest topographically organized cortex, like "early" vision, audition and tactile sensing, can be emulated using minicolumn models similar to the hybrid model we created, and that the emulation is computationally tractable on, for example, a small number (hundreds) of Cell Broadband Engine ® (Cell-BE) class chips. "Higher" cortical regions, because of plasticity needs, may require more computationally intense models, which deal with spiking dynamics and liquid state machine effects.

The effort has produced 8 publications (see Appendix 1) describing findings. It has produced emulations of neuromorphic computational models that make use of advanced computational architecture developments such as large scale multicore systems, large clusters of Cell-BE processors, Field Programmable Gate Arrays (FPGA), and other large scale parallel systems. It has also produced a series of briefings, which provide prerequisite background useful for exploring neuromorphic computing. These include briefings covering topics in neurobiology, the visual tract, cortical models based on Bayesian trees, on arrays of attractors, laminar columnar models on the neuron level, the Confabulation model, liquid state machines, and dynamical spiky neuron models.

Columnar model software produced by this effort includes C code and FPGA VHDL code. The VHDL code consists of "accelerations" of Bayesian tree recall algorithms, and BSB point attractors. The C code consists of these algorithms, models of minicolumns and functional columns, spiky neuron models, and Confabulation models.

This effort has produced some infrastructure suitable for continuing cortical modeling research. It consists of software, in addition to the models discussed, developed for and applied to modeling a visual input stream (a retina model, an optic chiasm model, and a thalamic-LGN model), a high throughput Publish/Subscribe messaging system, and high performance machine clusters (AFRL/RI's cluster of 288 PS3 CELL-BE platforms with 12 dual quad Xeon head nodes).

# 2. Background

This is a report on the work sponsored by the Air Force Office of Scientific Research and conducted at the Information Directorate of the Air Force Research Laboratory to investigate architectural issues surrounding neurobiological inspired computational methods based on networks of structures roughly emulating cortical columns. The work consisted of a three year multidisciplinary effort focusing on determining how neurological systems perform those aspects of cognition associated with sensing and perception aspects of cognition. The work has focused on ventral tract (object recognition) aspects of the brain. Dorsal tracts are parallel to ventral tracts, and are theoretically associated with spatial properties.

## 2.1 Rationale

The focus of interest is the development of computer architectures capable of advanced applications requiring large scale parallel computing and complex communication between nodes. The interest is driven by the value of applications which can make use of such architectures (perception and situational awareness are examples), and technology advances moving to surpass one thousand cores per die in the next few years. This technology trend is grounded in the problems of cooling, clock slew and parasitic inductances, which grow significantly as clock speeds increase. The previous rapid rate of clock speed increase for CPUs has disappeared. Chip developers have turned to multicore technology to make use of the continuing exponential trend towards increased transistor density. Multicore technology shifts problems from hardware to software and multiplies available parallelism. To make productive use of 100 thousand to 1 million processors, one must provide software, which can efficiently harness the parallelism inherent in the hardware. Software development is labor intense. The cost intensity grows significantly as parallelism increases. Software developers have few methods available to them to deal with parallel system design, except for messaging systems and multithread programming. No significantly better methods have emerged into common practice which displace or build on these. These techniques are suitable for small scale parallelism but grow unwieldy for systems even a few thousand processors. Existing High Performance Computer (HPC) platforms, like Blue Gene/L, can be configured with more than 130K processor cores. The challenge of harnessing parallelism on that scale for all but “embarrassingly parallel" applications (an application where very little communication is needed between processes) challenges the limits of programmability. Yet neural processing effectively harnesses parallelism on at least this scale.

Cognition presents as an excellent target of study because primate brains are examples of the kind of computing architecture we seek. It also holds promise to meet the “programmability challenge” of large scale parallelism with self supervised learning, and is therefore itself potentially a key technology for approaching other difficult to scale applications like Parallel Discrete Event Simulation (PDES). PDES applications are models of physical processes in terms of state changes at discrete points in time. Example PDES applications include networking, electronics, command and control, particle physics, machinery, weather, and communication systems. These applications are characteristically intense in terms of CPU but challenge computer architectures with the need to communicate events to all affected elements

within the simulation. PDES applications typically do not scale well across even a few hundred nodes. In parallel AFRL (6.2) research, new architectures are being developed to better address PDES, largely motivated by the neuromorphic insights uncovered in the research project. Specifically, the custom connectivity of synaptic networks is being mimicked in field programmable gate arrays to reduce the latency of event propagation across the massive architecture.

Beyond the rather pragmatic utility of PDES acceleration on neuromorphically inspired architectures, there are many aspects of cognition valuable to the Air Force missions which may be within near-term grasp. Some of these include learning, vision, audition and olfaction, ability to navigate an environment, and goal seeking.

These abilities have long been among the objectives of artificial intelligence research; but progress has been limited. In particular, solutions have lacked the robustness observed in natural systems. Thus competing "connectionist" approaches arise which draw inspiration from neurobiology to seek out these abilities. However, a significant impediment is that science has not yet worked out how the synapses, neurons and glia cells of a brain work systematically together to achieve cognition. Neuroscience has not traditionally been a "system centric" science. It tends to focus on ever narrowing details on anatomy, neurochemistry, and electrophysiology. While this narrowing of focus has progressed, related fields of interest have overlapped with neuroscience and provided resources crucial for the pursuit of neuromorphic computing. The medical community is an example; it relies on imaging technology for diagnosis. Neural imaging techniques are vital to neural surgeons and neurologists. Medical market pressures continue to promote improvement in resolution, and content. For example, the market has produced functional magnetic resonance imaging (fMRI) technology that can resolve blood oxygen level dependent (BOLD) response on a sub-millimeter level, providing a means for imaging the routing of cortical nerve bundles previously undetectable in living specimens. These kinds of advances, combined with emerging large scale computer technology, enable a new approach to investigating how brains work. The new capability is the emulation of large pieces of a brain.

It is becoming feasible to emulate full scale brains on a neuron level, at least insofar as computational complexity matters. The human brain has an estimated $10^{11}$ neurons, each with an average estimated $10^4$ connections to other neurons. Single neuron models need to account for synapses (connections) and somas (cell bodies). A simple synapse model uses two numerical operations (OPs): an index (address addition) and a value addition (this would be the complexity floor). A simple soma model (threshold compare and assignment) is equivalent to two OPs. Thus, a human brain emulation (if all neurons and synapses happen to fire at once; an unlikely event) would require ~$3X10^{15}$ OPs. A single Cell-BE node can peak at $2X10^{11}$ FLOPS. 15K such devices, by this measure, would be able to emulate a full sized human brain at about 1/1000 real-time speed. Certainly, synapse and neuron level models can be more complex than this estimate, but it is also true that emulation may not always need to be carried out at a low level. Moreover, it is often the case that one neuron connects with another multiple times, a situation that can be simplified in emulation by allowing for a "wider" weight range.

## 2.2 Basics of Brain Architecture

This brief introduction to neuroscience is intended to provide the reader with sufficient background for following the research presented in this report. More introductory details can be found online in [1] and [2].

### 2.2.1 Gross anatomy

The human brain consists of two nearly symmetric hemispheres divided front to back. It is also layered top to bottom. At the top is a sheet of tissue called the neocortex (often referred to as just "cortex"); it is roughly 2.5 square feet in area and 2 to 6 millimeters thick. Figure 1 illustrates a right brain hemisphere. The neocortex is the most prominent feature in the figure,

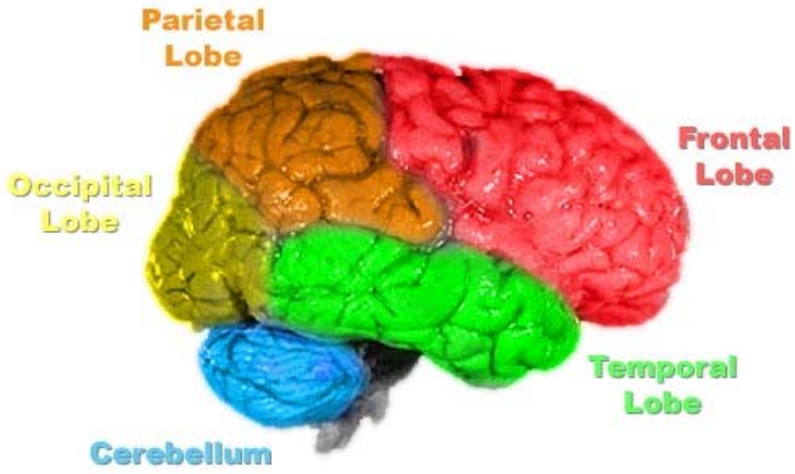



**Figure 1: Right hemispheric gross topographical view of a brain.**

and is depicted as four lobes: frontal, parietal, temporal and occipital. Beneath the cortex, the cerebellum is visible, and below that (gray) is the brain stem.

Figure 2 has a view of a brain from the back, illustrating the two hemispheres sitting on top of the Cerebellum, and a "cut-away" through the center of a brain showing the left hemisphere internal structure. The corpus callosum is a dense communication network (200-250 million axons, which are protrusions of neurons which carry output signals) connecting the hemispheres.

[1] http://commons.wikimedia.org/wiki/Image:Brain-anatomy.jpg

[2] http://www.mayoclinic.com/health/brain/BN00033

Note the location of the thalamus; it sits between parts of the brain involved in sensory interface (spinal cord / brain stem) and the cortex. Cognition is thought to be a phenomena primarily occurring in the neocortex, and modulated by interactions with the senses through thalamic connections. These interactions are sometimes referred to as thalamocortical loops, and occur along communication pathways called thalamocortical tracts. The tracts are bundles of axons (white matter), each axon a protrusion from a neuron, which form communication pathways.

Figures 1 and 2 depict the neocortex as a lumpy tissue, folded in on its self. When removed from the brain and flattened where it can be observed as a thin sheet 2 to 6 mm thick. These architectural features give the cortex a three dimensional character and are functionally significant. Absent in rodents and present in primates, the folds are referred to as sulci (or fissures), and the lumps between the folds as gyri (crests, or arches). At times, adjacent folds merge at the adjacency; the hippocampus is an example of such a structure. The folds and crests are often used as anatomical markers to locate functional areas.

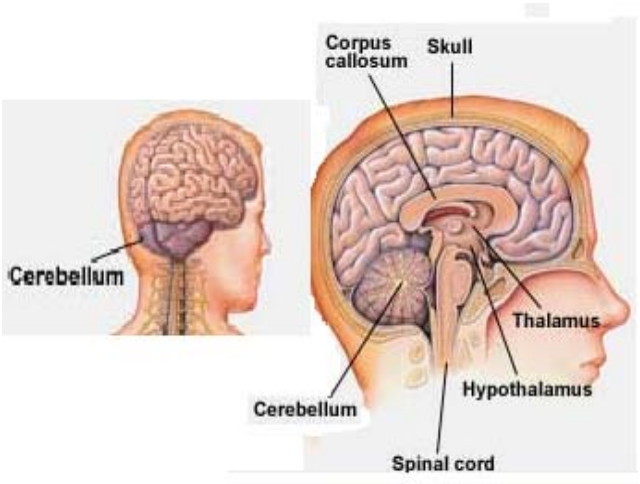



**Figure 2: Internal brain view.**

The cortex has been anatomically subdivided into regions associated with distinct functions. The traditional cortex subdivisions are the Brodmann regions [63], defined and numbered by Korbinian Brodmann based on Nissl (see http://en.wikipedia.org/wiki/Franz_Nissl) staining of cortical tissue. Brodmann's study revealed variations in cell structure of cortex tissue. The original study was published in 1909; it reported 52 areas thought to have distinct functions. Historically, the data on which function was associated with Brodmann areas was based on injury. Changes to behavior and abilities in subjects experiencing brain injury were attributed to the damage observed. Since then, the Brodmann areas have been refined mostly by subdivision. Advances in surgical methods (direct brain stimulation) and imaging technology, notably (functional magnetic *resonance imaging* (fMRI) and positron emission tomography (PET), have provided means to refine the mapping of function to cortical area. Brodmann regions continue to serve as a widely used method of identifying brain regions even though much more specificity has been achieved. Typically, investigators rely on referring to specific regions in reference to Brodmann areas (or regions).

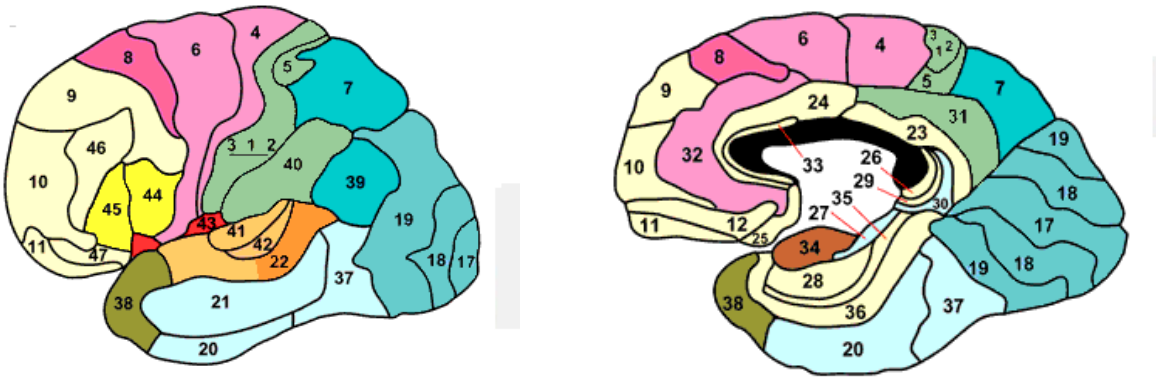



**Figure 3: Brodmann areas.**

### 2.2.2 Tracts

The brain is not a randomly connected population of neurons. Its connections are specialized to provide its ability to sense, perceive, store and process information, and direct behavior. Information enters the cortex in sensory fields such as: vision, Brodmann 17; and audition: Brodmann 41, 42. These primary fields translate sensation into a (in these cases) topographical perception. For example, area 17 may translate the rough equivalent of pixels into simple "percepts" like lines at various angles (orientation lines). These areas then connect to adjacent areas, which perform more translation (perhaps recognition of simple shapes). Within a few "hops" the degree of abstraction keeps increasing and the degree of topographical connectivity decreases. These connectivity chains are referred to as "tracts." Tracts are mapped through tracing dyes injected into neurons ( for example, in an animal, a colorant can be attached to a rabies virus and the virus injected into a neuron, and the virus will infect along the connectivity tract) through electrical probing to trace activity chains, and through metabolic imagery technologies such as fMRI. The techniques are expensive to apply, take a great deal of skill, and lack scale and precision critical to knowing details of connectivity sufficient to reveal fundamental computational methods. The strategies, which are best at detail, also cause harm to the test subjects and are limited by ethical standards. Notwithstanding these limitations, progress on non-invasive mapping of the physical connectivity and dynamical properties of tracts is accelerating due to advances in imaging technology driven by medicine. An example of this is Diffusion Tensor Imaging (DTI), a variant of fMRI, which detects water diffusion. DTI is routinely applied at medical centers such as the University of Pittsburg to map out white matter routing at a 1.4 mm resolution prior to brain surgeries. A 3D map assists surgeons with avoiding damage to those tract connections.

### 2.2.3 Neocortex laminar anatomy

The neocortex itself has six layers of cells. Roman numerals are commonly used for labeling the layers. Figure 4 is a stained neocortex cross-section specimen shown under magnification. The layers are labeled. This particular specimen is a slice from Brodmann area 17, the striate cortex. Area 17 is the primary visual cortex, or "V1". Initially observed by Brodmann, it is called the striate cortex because its layer IV is rather "fat," with layers (or striates) of its own. These sub-layers are labeled IVA, IVB, IVCa and IVCb. This broad IV layer is an artifact of V1 because of the massive interface between the nerves of the eyes (actually, at this point, relayed through the thalamus) and the brain. The eye information enters the brain at this layer, in V1. Layer I is the innermost region, and IV is outermost (adjacent to the pial surface). Layer I is primarily "white matter," or mylinated nerve fibers (axons) connecting neurons. Mylination is a covering promoted by glia cells. More than 90% of the cells in a brain are glia; about 10% are neurons. Glia forms the "life support" environment of the neurons. Mylination affects signal conduction speed in axons. The mylinated fibers typically extend further than unmylinated ones.

### 2.2.4 Neocortical Neurons

Neurons and their synapses are well accepted as the basic functional components of a brain, just as switches are the basic component of a digital computer. Neurons have inputs, called dendrites, and outputs called axons. Dendrites are characteristically short (typically 4 mm or less); axons can be short or long, depending on the morphology of the cell it protrudes from. The longest axons in human bodies are spinal. They may be up to a meter long. Neurons form circuits by connecting up dendrites with axons at junctures called synapses. Neurons produce electrical signals along these pathways. The signals may either excite or inhibit down-stream neurons. Neurons populating layers II, III, IV and V, are more involved with forming local circuits than those in I and VI. The specifics of the local and long range projections are not well known because of limitations of technology, and because a very large number of neurons in a brain would need detailed circuit mapping to distinguish circuit characteristics throughout the cortex. Aside from these limitations, there is a significant understanding, which provides a sort of "tool box of computation" from which hypotheses can be built for explaining how the brain may perform its work. For example:

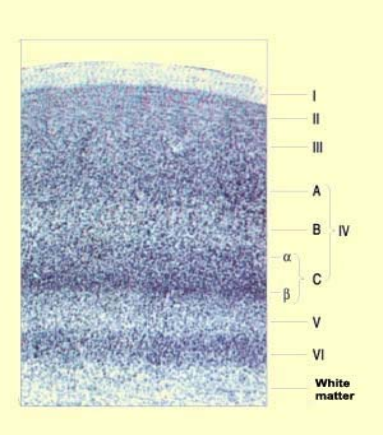



**Figure 4: Nissl stained cortex reveals layers and striations.**

- Neurons vary in morphology (size, shape) and electro-chemical characteristics throughout the cortex, and seem to be strategically placed within the layers to form circuits.

- Neuron electrical characteristics (electro-physiology) include the ability to send a single pulse, a pulse train spurt, or a long term spurt of pulses. There are differences in pulse frequencies. The transfer functions have been intensely studied and simulated [3].
- A single neuron is directly influenced by many other neurons. Neuron stimulation (of an individual cell) occurs at hundreds, and often thousands of synapses. A neuron's output is likewise potentially wide spread.
- The nature of a synapse can be inhibitory or excitatory. About 20% of cortical neurons are inhibitory [46].
- Neurons tend to be organized into columns, called cortical columns, perpendicular to the plane of the cortex. Columns in human brains tend to have about 80 to 100 neurons, except in the primary visual cortex, which are typically 150 to 200 (because of the thick layer IV) [46].
- Columns, at least in some parts of the cortex, form larger collectives which seem to have functional distinctions [46].
- Connectivity between neurons is mostly local, and between larger collectives is likewise mostly local. There is evidence of a "small world" connectivity architecture. A small world network is one in which nodes are not connected to all their neighbors, but where any node can reach any other node in just a few "hops." The time needed for a signal to travel is remarkably consistent at all scales; independent of transmission length (mylination thickens on long range projections, and speeds up transmission) [5].
- Millisecond level synchronization of collections of neurons has been observed [53]. There is evidence of a nesting of recurrent circuits. Within a column, levels II/III may be tightly coupled and the likelihood of recurrent connectivity is increased because of relative high cell population and the characteristically small cells found there. There is a loop feeding primarily up from level I to level II/III, and then back down toward V. There are recurrences between nearby columns, and long range cortex to cortex (corticocortical) as well as cortico-thalamic in scope [13].
- Feed forward pathways between regions of the cortex tend to project from layer III, and terminate in layer IV. Feedback connections tend to project from VI and III and terminate in I [46].
- Connections are weighted (strength of connection) and the weighting is plastic (changeable). A connection strengthens when a sending neuron pulses to a receiver within a small window of time associated with the receiver pulsing (coincidence strengthens, non-coincidence can diminish) [53].

- Information supplied to the brain by visual, tactile and auditory senses is highly organized topographically and maps topographically to the senses. For example, an array of probes in a primate primary visual cortex will measure an "A" pattern of excitement on the tissue when a high contrast "A" is within the subject's visual field [12] (See Figure 5). The topographical pattern of activation in cortical fields diminishes with the distance from the primary field, and is replaced by a pattern of activation associated with features. This is consistent, to a degree, with a hierarchical organization. Each layer of cortex in a (ventral) sensory tract seems to refine object classification [54].

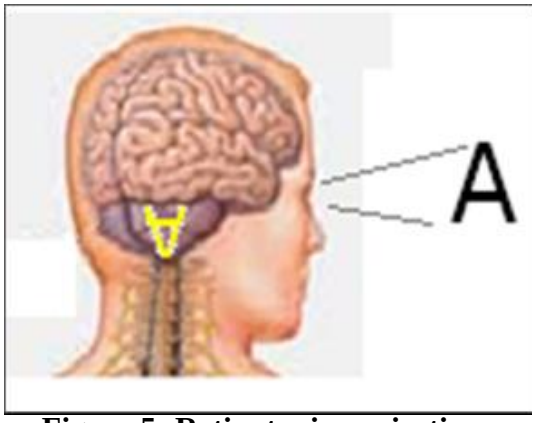



**Figure 5: Retinotopic projection.**

## 2.3 The Cortical Column Hypothesis

The search for evidence of how the parts of a brain work as a system to produce the phenomena of mind has been conducted on a variety of scales. The most detailed searches deal with dynamical models of individual cells connected together to form small patches of cortical layers, and even cortical columns. The most commonly known is the Blue Brain project being conducted in Europe. Blue Brain is reported to be an attempt to reverse engineer a whole brain. It seeks to replicate a brain at the neuron level. It deals with the challenge of emulating $10^{11}$ neurons, each with an average of $10^4$ connections in a three dimensional space for a grand total of $10^{15}$ connections. At this level of detail, the computational challenge is enormous. In 2007, the project reported a $10^4$ neuron model incorporating $3*10^7$ synapses (see: http://www.andyross.net/blue_brain.htm).

The Blue Brain project is expected to provide insightful descriptions of the mechanisms with which large scale neuron systems are assembled. The opposite extreme in the search are attempts to model cognition as behavior. The behavioral modeling approach provides insight into what the brain does (what the mind is), but it is not focused on how it does it. These kinds of efforts are traditionally symbolic in nature (as opposed to connectivist in nature) and rely on behavioral hypothesis (executive control, attention, and memory models) based on psychologically observable phenomena attributable to brain areas.

In between these extremes is modeling on the cortical column level, which spans scales from individual neurons to collections of tens of thousands of neurons. According to the hypothesis [46], the cortex is modularized into regions of varying size wherein the internal structure, function and external connectivity are similar, and that these regions of similarity are assemblies of cortical columns which represent another level of modularity because of how they connect. In this way a cortical column may be an architectural building block useful for understanding function. There are about $10^8$ cortical minicolumns in a human neocortex, and because these columns are much larger than a neuron (about 100 neurons to a minicolumn, a cylinder about 0.03 mm in diameter and 2 to 6 mm long), connectivity is significantly less than the estimated $10^4$ connections per neuron. For example, a minicolumn communicates with an area around itself with a radius of about 3mm. There are about 10K minicolumns in this area, and so connectivity between minicolumns is still 1:10K. However, there is only one entity, the minicolumn, instead of 100 neurons. Minicolumn connectivity is still incompletely understood, but there are clues which suggest the connectivity may be simplified by abstraction. There is evidence that the connectivity is sparse within this 3mm radius pool [5]. Perhaps 32 axons extend to about 32 minicolumns each, for a total of $10^3$ connections. If this estimate is correct, the grand total minicolumn level lateral connectivity, whole neo-cortex, is on the scale of $10^8$ X $10^3$ = ($10^{11}$) compared to the neuron scale of $10^{14}$. Long range projections are more difficult to account for, but are likely to be no greater than the local connectivity.

Connectivity complexity reduction is supportive of the argument that cortical column hypothesis provides good middle ground for exploring brain computational architecture. The computational architecture is expected to be diverse. It is not likely that minicolumns throughout a brain are functionally very similar. Differences in neuron population size and type, as well as differences in the glia environments surrounding them, are likely to account for distinctive field capabilities attributable to the minicolumns throughout the neocortex.

# 3. Research Objectives

The proposed effort lists three objectives:

A. Investigate alternative cortical column models. Specifically, investigate models proposed by Hawkins (a Bayesian model), and Anderson (a network of point attractors), and then develop and assess new models based on the work of these investigators and on other information acquired during a literature review. We wished to quantify how well the models perform cognitively, and the scale to which practical cortical models can be made. One expectation was to be able to describe plausible functional characteristics of biological cortical columns, and then classify computational cortical column models by comparison. Another was to be able to anticipate scales (cortical region sizes) which cortical model simulations will have to support in order to do productive simulator based research.
B. Evaluate the performance of large scale cortex models by simulation and emulation. This objective focused on developing experiments to assess models of regions of cortex. Regarding scale, one interest was to determine test bed capacity in terms of the pragmatics of how large a cortical region can be assessed on computational facilities at hand, such as conventional HPC clusters and platforms with FPGA augmentations. Pragmatics were grounded in the reasonableness of test case turn-around time.
C. Provide well defined and community vetted benchmark tests and metrics for evaluating simulated cortical models. These tests were to supply information for meeting the other two objectives. The benchmarks were to target specific facets of cognition efficacy, as well as "throughput" and speeds associated with simulation platforms.

Objectives A and B were focused to specific brain region areas to model as the result of neuroscience review. Our literature review led us to conclude the science best supported the modeling of neocortical sensing and perception areas, and the hippocampus. The most complete hippocampus work is based on rodent brains; sensing and perception has a major inclusion of human and near human (macaque) data. The hippocampus is located between sensory cortex and memory/association cortex within the temporal lobes. Its characteristics have been studied as feed forward transfer functions, and artificial hippocampus implanting in rodents has recently been achieved. However, from a systems perspective, it is not clear how these transfer functions might provide insight to the rest of brain computation. Context of what is happening on either side of the hippocampus was not clear. Early on, it seemed that the most productive approach would be to use information available on sensory cortex and make use of information from other cortical areas to bound hypothesis needed for "gap-fill". At this level, at least the sensory input context is well understood and much is known about primary sensory neocortical fields and the fields just beyond them – both in the neuroscience and behavioral aspects. Thus sensory/perception areas of cognition became the focus of the investigation.

Objective C was dependent on leveraging the work of DARPA ACIP and BICA efforts, both of which were very large efforts. It was hoped these efforts would provide testing environments with stimulation standards and data collection methods. In both cases termination occurred prior to the production of any of this. In response, objective C was redefined to identify candidate benchmarks which could be developed from the work associated with objectives A and B. We look forward to leveraging results from the SYNAPSE project on future work.

# 4. Task Performance

## 4.1 Task 1: Investigation of Alternative Columnar Models

Five activities were performed:

1. A literature review
2. An investigation of the hierarchical Bayesian model of invariant pattern recognition reported by George and Hawkins (2005);
3. An investigation of attractor network models;
4. An investigation of a Spiking neuron columnar model
5. An investigation of Confabulation
6. Development of a model to analyze on a large scale

### 4.1.1 A literature review

One of the objectives of this research has been to identify computational mechanisms used by the brain. A significant list of neuron based computational capabilities have been identified through examining electro physiologies of neurons, their morphologies, and through examining neural network configurations for operations such as filters, adders, and difference mechanisms [3]. Still other computational methods have been hypothesized by observing emergence of synchronization phenomena in otherwise apparently chaotic systems of neurons. These observations have been made in actual tissue and in simulations, and act on groups of cells on a millisecond scale. These oscillatory phenomena include notions referred to as synfire chains and polychronous groups [31, 53]. Cerebral computations which might be associated with chaotic system properties are not yet understood, but notions of "liquid state machines" are hypothesized [52]. Computational mechanisms associated with columnar architecture can include all of the above, but the most prominent architecturally based features are those associated with the interconnectivity of columns and the notion of nested recurrence of connectivity. At the center of these features is the notion of attractor networks, which themselves may be nested within larger neural nets. The nested recurrence provides an environment supportive of activity synchronization phenomena similar to what is seen in fMRI studies. Studies of how these phenomena are produced within interconnectivity schemes can be guided by cell morphology. For example, Johansson and Lansner have produced a columnar attractor model based on neuron level morphology and electrophysiology [34].

The science supporting the reverse engineering of a brain is significantly incomplete but expanding rapidly. Attempting to understand how a brain produces cognition by examining neuron based computation at or below columnar level is like attempting to understand how a program running on a digital platform like a LINUX cluster works by studying basic gates from which it is assembled. There does not seem to be a single dominant scientific discipline organized specifically for studying how brains compute; the community of investigators is

growing quickly, but it is a multidisciplinary group composed of engineers, neuroscientists, cognitive psychologists and others [4, 55]. Much of the funding for related research is based on health interests; especially the National Institute Health (NIH) institutes: NIBIB, NIDCD, NIGMS, NIMH, NINDS, and the pharmacology industry. The Blue Brain Project is currently the most focused neuromorphic reverse engineering effort. Its project director is a neuroscientist (Henry Markram), its manager is a physicist (Felix Schürmann), and the manager for Computational Neuroscience is Sean Hill, a Computational Neuroscientist. Blue Brain is primarily a bottom-up effort (starting with a molecular foundation) to discover computational mechanisms and organization.

Clearly, there is a need to study systems built of these components, but we have only scant descriptions of neuronal assemblies. This want of information forces investigations to proceed with hypothetical assemblies, which may become less hypothetical as bottom up efforts fill in details. We wish to use as high a level of abstraction as is practical to study how systems can be assembled to produce cognitive phenomena, while maintaining a strong neuromorphic foundation.

The cognitive phenomena we want to produce with these assemblies are problematic. Cognitive phenomena are, like their neuronal scaffolding, only sparsely understood. Cognitive phenomena investigation has been based on observable behaviors, and so it tends to be a "macro-phenomena." Attempts to relate behavior observations to brain areas are dependent on trauma data, animal studies, and technology limitations of non-invasive testing. These restrictions have limited progress on relating behavioral phenomena to cortical structures and therefore the minimum scale at which cognitive phenomena may be attributable to neuronal assemblies. In general, the lower limit of areas of the neocortex that have been associated with distinct function roughly corresponds to large fractions of the Brodmann areas in scale. It has also been observed that the neocortex tissue juxtaposition is related to hierarchic interactions, and observed function seems to be consistent with a hierarchical processing [17]. It is therefore argued, that "gap-filling" discovery can be successful when interactions of multiple cortical fields engaged in hierarchical interaction are conducted through emulation.

The idea that interactions of multiple cortical fields need to be represented for emulation studies brings forth the problem of where to start. The primary visual cortex (V1, area 17) was selected for this investigation. The rationale is based on evidence that fundamental functions of the visual cortex are in place prior to learning experience [22]. This "pre-wiring" suggests no learning is necessary to acquire interesting effects. Moreover, there is good anatomical data available for mapping afferent (input) connections in V1 coming from the eyes, through the thalamus, making a connectivity estimate practical. V1 has been studied extensively. The major information gaps include details of afferent connections to cells within minicolumns, lateral connectivity between minicolumns, and connectivity to other than V2. The role of cytochrome oxidase blobs (artifacts roughly in the center of assemblies of about 65 minicolumns known as functional columns) is not clear enough to define a meaningful functional role for them, but they are known to be associated with color perception. Pinwheels (areas of reduced distance between orientation columns) are likewise unaccounted for, though recent reports suggest pinwheels are related to complex cell orientation and direction connections to simple cells [61].

An estimate of computational complexity of full scale V1 emulation was made to look at the feasibility of full scale modeling of cortical fields. The estimate was based on representing minicolumns as "integrate and fire neuron" models. This kind of neuron scale emulation is thought to be more computationally demanding than more abstract, less neuron based models, and thus serve as a conservative estimate. However, it is far simpler than a spiking dynamical model and, as it stands, does not account for many dynamical characteristics of neurons.

The "integrate and fire" neuron model is the summation of the synapses of a neuron, and then subjected to a threshold function. The synapse summation is a weighed summation, equivalent to a dot product between a weight vector and a neuron value vector. There are about 180 neurons in a V1 minicolumn, and perhaps 30 of them are tightly recurrent within a minicolumn. These would be connected to fewer neurons than the others and we placed that estimate at 100. The other 150 neurons are assumed to be connected to about 1000 neurons.

Dot product complexity: 2N (N multiply, or Add operations).

Total = 2N (1000) + 2M (100), N = 150, M = 30
= 306,000 FLOPs.

To accommodate communication between minicolumns, we make the assumption that these neurons would cycle typically 5 times per saccade (a saccade is a rapid eye movement)

5X306,000 = 1,530,000 FLOPs.

There are about 5 saccades/second: 7,650,000 FLOPS/Minicolumn/Second

There are about 1.6 million minicolumns/V1.

1,600,000 X 7,650,000 = 12.16 TFLOPS

V1 is about 1% of the entire neocortex, but the neuron density in V1, based on minicolumn characteristics, is about twice the density beyond V1. A rough estimate of whole neocortex complexity is:

50X12.16 TFLOPS (608 TFLOPS)

By comparison, the NRL XD1 platform (864 AMD Opteron processors, 2.2 GHz) is capable of 2.5 TFLOPS, peak, without the use of its FPGA coprocessors. This platform was one of the largest available to the team at the time this investigation was starting. This estimate is suggestive of a need to produce an abstraction of minicolumns at least X100 less complex than the neuron scale emulation, to support large scale emulation studies of just a few fields. The dependency on scaling will diminish as multicore parallelism scales up to bring down the cost per TFLOP. For example, AFRL/RI recently assembled a 288 node CELL-BE array using SONY PlayStation® 3 technology. The cost was roughly $400/node, and computational capability is approximately 44 TFLOPS peak.

It was decided, as a first approximation, that "pre-wiring" characteristics of V1 might diminish dependence on the spike timing synaptic plasticity that is the central feature of the spiky models. Not using a spiking model eliminates the need to clock emulation at sub-millisecond rates required by spiking models [31]. It also reduces the opportunity to study mechanisms of cognition related to synchronization of chaotic behavior, like polychronous groups and synfire chains, but it does not eliminate them. The synchronization would occur at columnar scales, and clock rates closer to input frame rates than electrophysiology emulation rates. This puts modeled synchronization phenomena at resolutions still far better than currently available fMRI BOLD response and Diffusion Tensor technology can detect, thus not compromising the usability of clinical data.

The most evident characteristics of the V1 pre-wiring is orientation sensitivity; this is the ability of V1 to detect contrast lines in the visual field at various angles, and the directions of the contrast lines (light to dark, dark to light). Part of the cortical mechanism for this is the use of "simple cells," which spread their receptive fields in oblong shapes, and at consistent angles, across a small aperture of the field of view. This circuit is not locally recurrent. These are traditionally modeled as Gabor functions, linear filters combining a Fourier filter and a Gaussian kernel. They are (biologically) numerous, with variations in scale and location within the FOV. These variances are thought to contribute to invariant feature recognition.

Assessment of V1 column efficacy was limited to detection of lines of orientation, and their direction. Each columnar model was thus considered, except for Confabulation. Two types of attractors were investigated for orientation line detection efficacy using (feed forward), and expectation handling (feed-back): Brain State in a Box (BSB) [2, 58]. Preliminary assessment of the attractors suggested these attractors were useful for recognizing features using feed forward (afferent) data as well as feedback (expectation) data. They are also arguably closely neuromorphic, since they are both examples of recurrently wired integrate and fire neural nets. The BSB model utilizes real numbers within a range of [-1.0... +1.0]; in contrast, the Willshaw [58] attractor is binary. In our testing the BSB attractor was overall more able to converge to a basin than was the Willshaw, when many state vector element values were noisy (principally because of the differences in element value range). The Bayesian network approach reported by George, Hawkins was considered as well but that model is not as neuromorphic in nature as the attractors, based on the tightly coupled three layer tree structure [15]. The Bayesian model attempted to recognize more complex features than orientation lines, and thus be equivalent to multiple layers of minicolumns. It was able to recognize test data published by the George and Hawkins at about the same level of success they reported; about 50%. It was not considered for larger scale testing because it was not closely neuromorphic, and because of difficulties with training. However, Bayesian networks have been successfully exploited in the decision making applications. For this reason, it may be useful, in the future, for modeling "higher cortical regions," serving as an environment for assessment of collections of sensory fields. With this in mind, a three layer Bayesian network was examined for its ability to be sped-up using hardware acceleration (FPGA porting) and is presently being examined for porting to CELL-BE platforms.

Confabulation was reported as potentially useful for any feature parsing but demonstrated only for textual stream sentence processing [17]. The algorithm is computationally similar to Bayesian Belief, but it does not use a Belief tree network. We decided to explore Confabulation first in its reported context (textual data) and to consider it later on as a candidate for extra striate (above V1) modeling, fulfilling an expectation role.

### 4.1.2 Hierarchical Bayesian model of invariant pattern recognition

In his book **On Intelligence**, Jeff Hawkins proposes a description of human cognition based on the work of Lee and Mumford [39], and Pearl [48]. He argues that, given the main part of the brain dealing with learning and cognition is the neocortex, the neocortex functions though matching input sensory patterns with stored memories. Since it can recognize the same pattern in various forms (such as a face under different lighting conditions) the neocortex can be described as an invariant pattern matching device. Dileep George and Jeff Hawkins developed a model of the visual cortex based on these properties of the neocortex. The model is centered around a Bayesian inference network where all levels in the hierarchy use the same computational rules. The model was implemented in MATLAB and was capable of recognizing a set of images under various transformations.

Our interest was focused on how this model maps to cortical columns, in a neuromorphic sense; whether the reported spatial invariant object recognition results were reproducible; and how the model scales to full neocortex scale. Dr. Tarek Taha, Clemson University, and two of his graduate students performed this investigation over two summers while resident at RRS. The George/Hawkins algorithm contained training and recall operations, but training was not tackled during this investigation. The "recall" part of this algorithm, based on a published MATLAB implementation, was re-implemented using C language, and then highly optimized. The final version of this reported algorithm was cast in fixed-point arithmetic. It was run on a 3 GHz Pentium 4 platform to verify recall efficacy, and to serve as a performance (processing time) baseline.

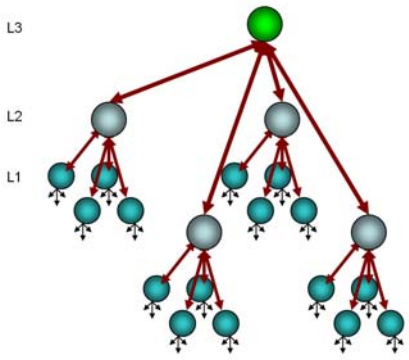



**Figure 6: A simplified model of the Bayesian network in the George and Hawkins model.**

The reference algorithm consisted of a three layer network, with the lowest layer (L1) consisting of 64 nodes, the middle layer (L2) consisting of 16 nodes, and the top layer (L3) consisting of a single node (see Figure 6). Each node used only a single parent. The network was trained to recognize a set of 91 images. In the image, recognition phase the input image is fed to the layer 1 nodes. These nodes compute a local belief of which trained image is most likely to match the

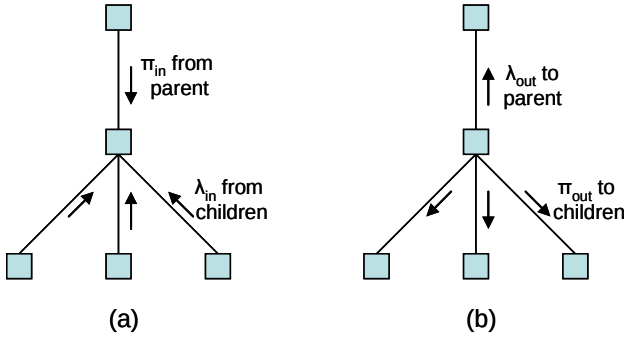



**Figure 7: Belief transfer in a Bayesian tree. Squares represent computation nodes. (a) Gathering beliefs from parent and children nodes before node computation. (b) Distribution of beliefs to parents and children nodes after node computation.**

input and propagate a set of values based on this belief to their parents. Since the model captures the uniform computation property from Hawkins' description of the neocortex, the same set of computations take place at all the nodes. The belief propagation models defined by Judea Pearl define these computations [48]. Each node processes input beliefs from its parent and children nodes ($\pi_{in}$ and $\lambda_{in}$ respectively as shown in Figure 7).

Once processing is complete, output beliefs are sent back to the parent and children nodes ($\lambda_{out}$ *to parent* and $\pi_{out}$ *to child*, respectively as shown in Figure 7b). Equations 1 through 6 are utilized in generating the beliefs. Each node has a unique training matrix ($P_{xu}$ in equation 2).

$$\lambda_{product}[i] = \prod_{child} \lambda_{in}[child][i] \quad (1)$$

$$F_{xu}[j][k] = \pi_{in}[j] \times P_{xu}[j][k] \times \lambda_{product}[k] \quad (2)$$

$$m_{row}[j] = max(m_{row}[j], F_{xu}[j][k]) \quad (3)$$

$$m_{col}[k] = max(m_{col}[k], F_{xu}[j][k]) \quad (4)$$

$$\lambda_{out}[j] = m_{row}[j] / \pi_{in}[j] \quad (5)$$

$$\pi_{out}[child][k] = m_{col}[k] / \lambda_{in}[child][k] \quad (6)$$

The MATLAB reference model produces beliefs, at the top node, of what image is being viewed at the bottom nodes. One image at a time is presented to the system. The images presented to the model are all simple line drawings (see Figure 8) imposed on a 32×32 pixel array. Each level 1 node viewed 4×4 pixels, and computed a belief based on it and "expectation" data from level 2. Each level 1 node presents to a level 2 parent a belief vector. Level 2 nodes, in turn, produce beliefs which get sent to the respective parent node, and back to child nodes.

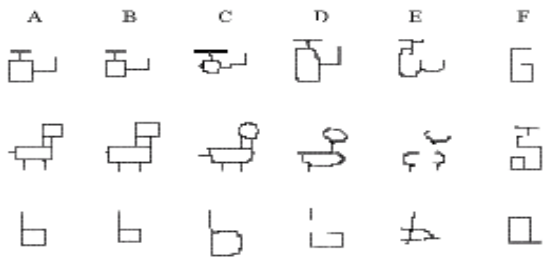



**Figure 8: Images typical of what was used as test data for the Hierarchical Bayesian model.**

Our experience with the algorithm's ability to recognize was consistent with the authors' report; about 50% recognition rate, and a spatial invariance for lateral displacements, but not rotational. The system had not been trained for rotational invariance (the invariance training involved many transitioned images). Numenta Inc., a company founded by Hawkins and George, is developing newer versions of this algorithm currently. The newer versions of the algorithm capture the temporal domain (in addition to the spatial domain considered by the initial version), and provide more sophisticated recognition. Implementations and descriptions of the newer version are available at Numenta Inc.'s website.

### 4.1.3 An investigation of network of attractors

AFRL/RI hosted a review of symbolist and connectivist research at Cornell University in July of 2005 with the intent of identifying technological gaps in cognitive computing. Presenters from AFOSR, multiple well vetted research institutions (MIT, Georgia Tech, Cornell, USC, Brown, U. Michigan, U. Mass, U. Binghamton) and industry (Saffron Tech., Lockheed, Raytheon, Lexxle, Aptima, Mgt. Sciences) presented their state of the art assessment and insights on gaps [45]. Among the speakers, Dr. James Anderson (Brown) presented his "Ersatz Brain Project" at that review. Erzatz Brain is an effort to model aspects of mind with nested networks of fixed point attractors. AFRL's review of Anderson's work was the basis of including an investigation of network of attractors as one of the "starting points" of task 1. After this effort commenced, AFRL awarded a SBIR phase II contract to Aptima, Inc., in conjunction with Dr. Anderson, to develop computing architectures based on the network of attractors idea. That final report is pending at this time.

Our interest was focused on how this bio-inspired model maps to cortical columns, in a neuromorphic sense; and how the model scales to full neocortical scale. Dr. Qing Wu, and Dr. Qinru Qiu, both summer faculty residents at RRS from Binghamton University, contributed significantly to FPGA and CELL-BE aspects of this task.

The Brain State in a Box (BSB) algorithm was selected as the attractor function to incorporate into the network study because of its association with the Ersatz Brain project [2]. BSB uses state vectors with "N" real numbers in the range of [-1.0...+1.0]. Its name is a metaphor for describing the algorithm as an N dimensional shape. Its fixed basin points of attraction lie in its corners. An N dimensional BSB function can separate M basin points, where M is ~15% of N.

The initial activities were to implement the Brain State in a Box (BSB) algorithm using the C language, characterize its complexity in terms of FLOPs (Floating Point Operations), measure its execution time on a conventional platform, and measure a sense of cognitive efficacy [2]. The algorithm itself is relatively simple:

$$X_{n+1} = S(\alpha W X_n + \lambda X_n + \gamma X_0);$$

X is a state vector of K elements.
$\alpha$ is a scalar constant.
W is a K by K weight matrix containing training data which defines the fixed point basins of attraction.
$\lambda$ is a scalar constant.
$\gamma$ is a scalar constant for maintaining stimulation (initial X)
S () is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1 & if \quad y \geq 1 \\ y & if \quad -1 < y < 1 \\ -1 & if \quad y \leq -1 \end{cases}$$

The algorithm accepts an arbitrary X and replaces it with an X' closer to a basin of attraction.
The numeric operation complexity (floating point operations) is $2K^2 + 4K$

The computational complexity increases with the square of the number of elements, and the yield of useful basins of attractions increases proportionately with K. Typically, multiple iteration cycles are needed to bring the BSB to a basin of attraction. The number needed is dependent on the number of elements in the state vector, and its initial value. Figure 9 illustrates how the complexity rises significantly as the number of iterations increases.

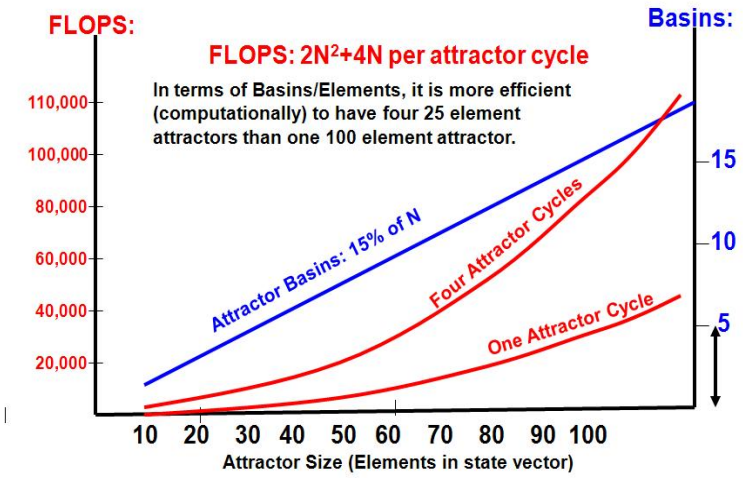



**Figure 9: BSB complexity versus basin availability**

The initial cognitive efficacy test explored how a BSB might close on a basin when it deals with partial data. The test assumed the BSB would be used simultaneously on multiple streams of input data corresponding to afferent data (sensory input), lateral data (effects of neighboring cortical columns in the same neocortical field) and expectation data (feedback from higher cortical fields).

An experiment was designed using a 19 element state vector with 16 elements tied to black and white pixel elements arranged as a 4X4 image. The remaining 3 elements of the BSB state vector were used to form a "tag" which could represent lateral or expectation feedback.

A vector of length 19 (floating point) elements was defined for the BSB. It was trained (100 times) using the following pattern:

**Table 1: Element number**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | 1 | 1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |

**Table 2: Element 3 through 18 represent a 4X4 pixel pattern**

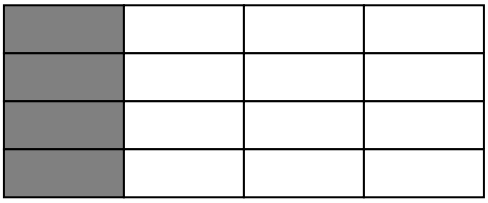
| 3 | 4 | 5 | 6 |
|---|---|---|---|
| 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 |

**Table 3: Element 3 through 18 values**

| | | | |
|---|---|---|---|
| 1 | -1 | -1 | -1 |
| 1 | -1 | -1 | -1 |
| 1 | -1 | -1 | -1 |
| 1 | -1 | -1 | -1 |

A vertical pattern we designate as VP1

**Table 4: 4X4 VP1 pixel view represented by this pattern**



If a "1" is considered energy in a pixel, and a "-1" considered no energy, then Table 4 depicts the pattern from Table 3 for the first vertical pattern. Patterns VP2, VP3 and VP4 correspond to the energy column being the second, third and forth columns, respectively. Four analogous horizontal patterns exist, as well as seven each for the two diagonal classes.

The first three elements [-1, -1, 1] are use for the "Tag." Think of Tag as a predisposition.
The neuromorphic basis of this experiment rests on the speculation that a minicolumn in the primary visual cortex has tightly connected recurrence of about 20 to 30 neurons and a small field of view (incoming occipital signal). Furthermore, that the minicolumn will have to somehow detect a visual contrast vertical line cast through the small field of view. A small collection of these might cooperate to recognize all vertical bar patterns of interest.

We expect a BSB, so trained, will be able to recognize its trained pattern reliably, and similar vertical patterns less reliably. Furthermore it is expected that patterns corresponding to horizontal and diagonal bars will be rejected; this "column" should be specific to its own and similar vertical patterns.

Unique tag patterns were defined for vertical, horizontal and diagonal lines (both back slash and forward slash diagonals). Training patterns for these were defined as well, but training was performed on one vertical pattern only (VP1). Training was performed for 100 cycles. No training was performed for VP2, VP3 or VP4 vertical patterns, or other patterns.

**Table 5: Arbitrarily assigned tags for each class**

| | | | |
|---|---|---|---|
| Vertical Class | -1.0 | -1.0 | +1.0 |
| Horizontal Class _ | +1.0 | -1.0 | -1.0 |
| Forward Slash / | -1.0 | +1.0 | +1.0 |
| Back Slash \ | -1.0 | +1.0 | -1.0 |

Recall was then attempted for each of the pattern classes (vertical, horizontal, and diagonals). Recall was attempted with tags set as:

- The correct tag for the class (strong Bias);
- A fraction of the correct tag for the class (scale factor, or "Small Bias")
- All zeros (No bias)

There are several methods of evaluating an associative match; Hamming distance and Euclidian distance are common. This experiment chose to use the tags themselves. The idea was that the tags might represent efferent projections of a column which extend laterally, backwards or forwards. The tags, like all elements of a pattern, are real numbers. One can select a radius close to an attractor corner to convert an element within a radius to the full corner position. In this case, we choose to require hitting a corner (zero radius) for success, but the data is available to determine how close each recall came to a corner.

Recall was cut off after six iterations.

Results showed that without expectation (tag field set to zero) the VP1 pattern was matched even when the magnitude of the incoming signal was ¼ full strength on each element. No match was possible on the other vertical patterns with null expectation, but they all matched with even low (1/4 strength) expectation. Otherwise the algorithm rejected (did not converge on a tag) when a pattern was not a vertical pattern, and a non-vertical tag was used for expectation but, there was drift toward the wrong corner. These findings supported there being a neuromorphic efficacy to be able to "conclude" a feature recognition based on incomplete data using a small number of cycles with a BSB on the scale of a minicolumn.

Similar exercises were conducted with BSB sizes of 32, and 128. The 32 element case used a similar number of cycles to come close to basins. The 128 element case required more than ten recursive cycles where clear but minimal signal was present. Too many cycles ran the chance of converging invalidly.

### 4.1.4 An investigation of spiking neuron columnar model

While the main theme of this three year effort is centered on investigating brain architecture modularity well above the neuron level, certain reports in the literature suggest sub-neuron level features may be important to cognition and also be feasible to emulate on a large scale. Two such features are neuron electrophysiology and Spike Timing Dependent Plasticity (STDSP). Electrophysiology has to do with neuron electro chemical dynamics; the spiking characteristics of these cells. STDSP has to do with the dependency of synaptic weights on millisecond level coincidence of pre-synaptic and post-synaptic firing. Eight man weeks were set aside to look into how computational mechanisms associated with these features can be investigated. During this time literature reviews were performed on Hodgkin/Huxley (HH), Morris-Lecar, and Izhikevich models, as well as literature on electrophysiology and morphology neuroscience. Software was built to investigate emulation strategies and to perhaps be able to produce some oscillatory effects reported by Izhikevich.

Alan Hodgkin and Andrew Huxley are credited with the first analytical models based on living neurons [21]. Their work was based on giant squid axons. They produced compartmental nonlinear ordinary differential equations approximating neuron transfer functions from one side of a compartment to another. Individual neurons can be described in terms of compartments by chaining compartments together. Each compartment‘s biophysical characteristics were represented by circuit analogs. Capacitance was used to approximate membrane separations, non-linear conductance to model voltage gated ion channels, and current sources to model ion pumps. (A good introduction to this topic is in Chapter 4 of the Book of Genesis, downloadable from: http://www.genesis-sim.org/GENESIS/bog/bog.html .)

In principle, any neuron can be represented by such compartmental models. Simulation tools have been developed and refined over the years (e.g. Genesis, Neuron) and have been funded by NIH grants and perhaps other sources. They are freely available, and have improved in terms of numbers of models placed into in databases, details of models, and scales of network models (multiple neurons, each compartmentally modeled). Many compartmental analytical models (differential equations) in addition to the original Hodgkin/Huxley (HH) model have been produced as well [32]. They are characteristically very computationally intense.

Izhikevich described a simplified model which he proposed for studying oscillatory synchronizing phenomena [31]. It uses a simplified single compartment model of a neuron soma and spike timing dependent synaptic plasticity (STDSP) to model synapses. His model cycled at 0.5 milliseconds (emulated clock rate) and reportedly produced sleeplike polychronous gamma oscillation patterns at 40 Hz. This report was used as a basis to investigate how minicolumns might be built using Izhikevich’s compartmental spiking model.

The objectives were to be able to measure conventional platform performance of an assembly of minicolumns numerically equivalent to a functional column, and to see whether polychronous activity can be readily produced from a functional model scale. (Another rationale for doing this is to become familiar with the information gaps inhibiting this line of research.) Izhikevich’s experiment used 1000 randomly connected neurons, 20% inhibitory and 80% excitatory. A 0.1% probability of connection between any two neurons was used.

In our study data from was used to estimate the cell populations in the layers of a prototypical V1 minicolumn (Figure 10) [46]. Data was macaque monkey data, and it was incomplete. For example, morphologies were not related to electro-physiologies, and detailed connectivity data was absent. The morphologies listed were limited. Based on the data available, a "rough model" was defined which related layers, morphologies, excitatory (Non GABAeric) and inhibitory (GABAergic) characteristics, and populations (numbers of). The rough model used 156 neurons, 5 morphologies, and four electrophysiologies.

| | % of ('156) neuron population | | Numbers of Morphology type | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Non GABAergic | GABAergic | AP | SStellate | LB | SB | Arcade | Ch | DB | Ng |
| | | | | | Big Guess | | | | | |
| I | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| II | 11.25 | 3.125 | 0 | 18 | 0 | 2 | 0 | 0 | 2 | 0 |
| III | 11.25 | 3.125 | 17 | 0 | 3 | 0 | 1 | 1 | 0 | 0 |
| IVA | 8.75 | 2.5 | 0 | 14 | 4 | 0 | 0 | 0 | 0 | 0 |
| IVB | 7.5 | 2.5 | 12 | 4 | 2 | 2 | 0 | 0 | 1 | 0 |
| IVCa | 8.75 | 1.25 | 2 | 8 | 0 | 0 | 0 | 1 | 0 | 1 |
| IVCb | 12.5 | 2.5 | 20 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| V | 7.5 | 2.5 | 12 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| VI | 12.5 | 2.5 | 20 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |

**Figure 10: Prototypical V1 column model.**
**156 neurons. Key: AP (Apical Pyramidal); SS (Spiny SStellate); LB (Large Basket); SB (Small Basket, Ch (Chandelier); DB (Double Basket)**

A connectivity estimate was made:

1. Double Bouquets in II project down into III, IV, V. Most dense in III is basis for the rest [10,61].
2. Magnocellular=> IVCa Spiny stellate, Blob; Parvocellular => IVCb, Interblob.
3. IVc projects tp II/III.
4. II/III projects to extrastriate V2, …Ventral.
5. IVb sends to dorsal extrastriate.
6. IVCa sends to nearly every cell in IVB.
7. IVCb non-spiny stellate send to IVB.
8. All V pyramidal cells send to IVB, but half of inhibs do also.
9. ½ of layer IV cells send to IVB.
10. Every layer recurrent within self.

An estimate was made for signal timing. The dimensions of a minicolumn are such that locally connected neurons are typically within a fraction of a millimeter with each other. Those that are at extreme ends of a minicolumn are no more than 3 millimeters apart, but those have *myelinated* connections. Neuron signal speed is in the range of 1 meter/sec to 100 meters/sec. If we just use the slowest number, that turns out to be about a 1 ms/mm delay. Spike Time Dependent Synaptic Plasticity has a window significantly wider than 1 ms. Therefore, using 1 ms delay for all connections within a minicolumn is probably a useful simplification. (A ½ ms clock is used by Izhekevich for polychronous group modeling.) We might choose to use N ms for lateral (neighborhood) extracolumnar connections, where N is the distance in minicolumn width between source and destination.

Electrophysiology was modeled using Izhikevich's equations [30]. Izhikevich's equations use 4 parameters to shape electrophysiology: a (time scale of recovery variable u), b (sensitivity to recovery variable u to sub threshold fluctuations of membrane potential), c (after-spike rest value of membrane potential, and d (after-spike reset of recovery variable u.) Three excitatory parameter sets were defined, and one inhibitory. The excitatory parameters were consistent with: regular spiking, intrinsically bursting, and chattering models. An inhibitory fast spiking parameter set was chosen.

**Table 6: Estimates of characteristics as a function of morphology**

| Morphology | Characteristic |
|---|---|
| Pyramidal | Excitatory |
| Stellate | Inhibitory or excitatory |
| Basket | Synaptically inhibitory |
| Chandelier | Perisomatically inhibitory |
| Double Basket | Perisomatically inhibitory |

There was an attempt to match the spiking models to morphology. No published reports of this relationship were found. It was as if the electro-physiologists and morphologists ignored each other, but both types of information are vital. The following estimates were made (see Table 6).

The difference between synaptic and perisomatic connections is that the perisomatic connections are very strong; they are where axons connect directly to somas instead of to dendritic synapses.

Connectivity rules within a minicolumn were:

1) Layer II
   A. Double Bouquets in II project down into III, IV, V. Most dense in III (80%) else 30%.
   B. Stellate cells in II project to all cells in II.
   C. Any other inhibitory cells in II project to hillocks of just one or two excitatory cells in II.
2) Layer III
   A. Apical pyramidals in III project into II, 30% connectivity, and within III, 80% connectivity.
   B. Inhibitory in III project to one or two pyramidals in same layer.
3) Layer IVA
   A. Excitatory neurons project to all neurons in this layer with 70% likelihood of connection.
   B. Inhibitory neurons project to each other, and excitatory neurons with a 40% likelihood.
   C. All neurons in this layer project into III with a 25% likelihood of connection.
4) Layer IVB
   A. All neurons in this layer project into III with a 25% likelihood of connection.
   B. All neurons in this layer connect with each other with a 60% likelihood.
5) Layer IVCa
   A. Excitatory neurons project to IVB pyramidals, stellate, and inhibitory (80%).
   B. All pyramidal neurons in this layer connect to layers III (40%) and II (25%).
   C. All neurons in this layer connect to all in layer, 80%.
6) Layer IVCb
   A. Excitatory neurons project to IVB pyramidals (80%).
   B. All pyramidal neurons in this layer connect to layer III (40%).
   C. Inhibitory cells connect to about 10% of this layer.
7) Layer V
   A. All pyramidal neurons in this layer connect to layers IVA (20%), IVCa (20%), IVCb (20%), III (60%) and II (20%).

The “final” minicolumn model made use of 125 neurons, and thus was smaller than a V1 minicolumn but larger than most other cortical minicolumns. 8,000 neurons were emulated.

A 64 minicolumn assembly was configured into a functional column, with minicolumn lateral connectivity based on cell morphology (plume diameters and distances). Two extra neurons were created to represent thalamic “drivers” clamped to a constant stimulation. These oscillated at about 5 Hz and drove several layer IV neurons.

Runs were performed for as long as one hour and an attempt made to find occurrences of polychronous groups as STDSP formed up the groups. The technique used was to keep track of neurons contributing to the firing of a neuron. The groups which formed oscillated with the thalamic neurons and tended to be invariant. It is assumed that the software created for the test is not yet dependable, and polychronous grouping results not yet valid. More work will be required to follow that interest, and may be taken up after other investigations report out on characteristics of oscillatory phenomena.

However, valuable information was derived from the study. We became aware of the gaps in understanding of minicolumn structure science, cell morphologies, and electrophysiology. It was observed that the model, which exhibits relatively high and complex connectivity (the STDSP model) can be performed in about 1.5% of real-time on 2 dual node 2.2 GHz platform, using C software. Most of the computational load was associated with the (sparse) STDSP connectivity and need for sub-millisecond clocking. Eight man weeks were put into this study, including the research on STDSP, morphologies and electro-physiologies, and software development. We estimate that reliable polychronous group data could be derived from test runs with another 8 weeks of effort. The software consists of 3624 lines of C code. The long emulation times, detailed models and need to bridge large gaps in understanding seem to put the spiking model path of columnar research into a high risk category that can be set aside until it is needed to solve problems that cannot be tackled with more abstract models. It is also true the Blue Brain project is filling these gaps. If the detailed columnar models they produce are published, it will be a good leveraging.

### 4.1.5 An investigation of confabulation

The literature review component of this task surfaced reports by Robert Hecht-Nielson of a cognitive mechanism which explains all of cognition [20]. The center piece of his reports both published and in presentations, was a demonstration of software which completed sentences with no context, and another which completed a sentence in the context of two other sentences. The evidence presented was impressive; correct or nearly correct grammar and sentences which made sense to the reader. The author was careful to point out that the illustrated results were carefully selected; that the technique was prone to a kind of "babble" commonly associated with persons experiencing significant cognitive deterioration, a condition clinically referred to as "confabulation."

The hypothesis is that the reported algorithm models the fundamental cognitive mechanism, and that the mechanism must be somehow layered on a large scale (many interconnected confabulators) to produce a level of coherence.

The reported sentence completion algorithm trained by reading text; lots of text. It then "recalled" by using a context (for example, the start of a sentence) to retrieve a sequence of words and phrases which its training statistically connected to the context. The training consisted of reading one sentence at a time and breaking it into sequences of words and phrases - all possible combinations of these. Sentence by sentence the training keeps track of all words and phrases encountered, and all sequences formed, through statistical links.

Confabulation was deemed worthy of detailed investigation. The following goals were established:

- Define, exactly, training and recall algorithms (the initial Hecht-Nielson descriptions were vague).
- Define appropriate data structures for efficient operations on the data.
- Determine metrics for measuring performance.
- Assess prospects for performance (speed) improvement. This was carried out in Task 2.

The following subsections organized as follows: a preliminary implementation and test of the confabulation model (4.1.5.1); an overview of the confabulation model and sentence completion application (4.1.5.2); details of the training and recall algorithms used in the model (4.1.5.3); appropriate data structures, and their effects on speed and memory usage (4.1.5.4); metrics to assess performance (4.1.5.5); and the prospects for speeding up and scaling the model in embedded hardware and distributed cluster forms (4.1.5.6).

#### *4.1.5.1 Preliminary implementation and test of the confabulation model*

Two forms of the same confabulation algorithm (sentence completion) were built to determine efficacy and baseline performance. The first used full vector matrix mechanics, although it was known that this form would be computationally intense because the vectors and matrices were very large. They were also thought to be sparse, but the degree of sparseness was not known and so a second sparse data version was developed as well. Trees were used for sparse data representation as a starting point, with the expectation of revisiting data structure design after experience with the algorithm provided some statistics on how the sparseness was organized.

Initially, the full matrix implementation required very large amounts of data space and executed very slowly. This limited tests to exercises using very small amounts of training data. The sparse data version seemed to execute at disk I/O limited speed and was able to easily absorb a great deal of text. Using limited training data, both versions were numerically compared as a verification step. Under task 2 (Section 4.2), the scalability of both versions was examined.

The sparse data version was used for efficacy assessment. There were no pre-existing efficacy measures for the confabulation model. We requested specific training corpus and test criteria data reported generally by the author (for verification of reported experiments) but it was not made available to us. An attempt was made to measure the success rate of the production of sentence completions which “make sense” but it quickly became clear there was high sensitivity to the corpus of training data, and the words and phrases used in the starter sentence. Instead, we settled for subjective measures at this point, and deferred quantitative measures for confabulation to later work (see Section 4.1.5.5).

We selected a corpus of texts for use in training experiments, with the intention to represent many cultures, times and styles. The material consisted of:

- Adam Bede, George Eliot. 1859
- Relativity: The Special and General Theory. Albert Einstein. 1920
- *Flappers and Philosophers,* F. Scott Fitzgerald, 1920
- This Side Of Paradise, F. Scott Fitzgerald, 1920
- Gettysberg Address, Abraham Lincoln, 1863
- A History of England, 6 volumes. Hume. (1754–1762).
- Differential Patterns of Effortful Control and Subsequent Problem Behaviors: An Examination of Peer Experiences as a Mediational Mechanism. Jessical Moore, 2007, UNCG Thesis Proposal.
- JFK Inaugral Address, John Kennedy. 1961.
- The Life of Johnson. (Complete). James Boswell, 1844.
- Laws. Plato. 360BC.
- Statesman. Plato. ~360BC.
- Symposium. Plato. ~360BC.
- The Republic. Plato. 360BC.
- Timaeus. Plato. ~360BC.
- Precepts of Lord Ptah-hotep. 2350 BC
- Address to the Irish Parliament. Tony Blair. 1998.
- The Waste Land. T.S. Elliot. 1922.
- War and Peace. Leo Tolstoy. 1869.
- Short History of Wales, by Owen M. Edwards. 1901.
- Cat in the Hat, Dr. Seuss. 1957
- And to Think I Saw It on Mulberry Street, Dr. Seuss. 1937.
- Green Eggs and Ham, Dr. Seuss. 1960

Nine authors were used to demonstrate examples: Eliot, Einstein, Fitzgerald, Lincoln, Hume, JFK, Plato, Boswell, and Tolstoy. The settings used for certain confabulation model parameters, and the results of training in terms of data base size and time are as follows:

L2 confabulator: words and phrases used.
Minimum phrase count set to 4.
Minimum phrase length set to 2.
Phrase length limited to 5.
Sentence length limited to 20.

Total sentences read: 283,355, total words read: 3,640,544
Total knowledge links: 8,365,568
On a hhpc2 dual Xeon headnode the following performance was observed:

- Training (CPU) time: 45 seconds.
- Training (Wall Clock) time: 45 seconds.
  Training speed: 6,296 Sentences/Second.

We did an experiment to test the confabulation recall algorithm by using it to complete a number of short arbitrary 'starter' sentences. The results are show in Table 7, were the "best" completed sentences, which typically appeared on the first or second attempt (successive completions using the same starter sentence can be different because the algorithm has a stochastic element). The results were typically not text strings appearing in the training text. However, sometimes they were exactly the same as sentences in the training text (e.g.: "That it is …" and "Even the …").

**Table 7: Selected sentence confabulations**

| | |
|---|---|
| I am | I am dear Sir Your most humble servant and state and I give not only health and salvation to the |
| Come now | Come now what was this duel about ? |
| Even the | Even the magistrates of the had stayed away - - not one of them appeared except Mr . |
| Finally the | Finally the amendment was negatived |
| For this purpose | For this purpose he said that the object of child and difficulties which difficulties were increased by the |
| For this | For this is the way of the nature of |
| France was | France was solicitous to weaken the power of Moorish empire and so promote her own designs of encroachment. |
| From this time | From this time up to the end of foundation when serious disturbances occurred at Khelat the state of Scinde was |
| Thank God | Thank God there is no difficulty in seeing that that perfect number of time fulfils the perfect year when all |
| That it is | That it is expedient that tithes and necessary compositions for tithes in Ireland should cease and be forever extinguished |
| Then he | Then he asked what was the state of the shipping interest |
| They were | They were followed by Wellington who took up |
| Think of the people | Think of the people and ask yourself whether the world is more likely to be a believer in the unity |
| This motion | This motion which was intended to bring the expenditure of 4th of February the possible scale consistent with the |
| To be | To be sure he is likely to get the he would vote for |

In general sentences completed by the confabulator were similar in style to those in the training material. This effect was magnified by training networks on material from only one style or author. For example, when trained only on material from Shakespeare, using a short, general starter sentence produced the following completion that was not in the training test, but reflects the style of Shakespeare.

Original: "Go to the forge with it then shape it I would not have things cool."
Starter: "Go to"
Completion: "Go to me at your convenient leisure and you shall know how I speed and the conclusion shall be "

Similar effects were seen with training limited to JFK's speech, the Dr. Seuss material, and religious material.

Conclusions drawn from this brief exercise were that the confabulation model clearly produced interesting results, and that it should be considered for further investigation. We also determined that it would likely scale well, based on our observation of near disk I/O limited training speed on the traditional Linux platform, and the regular communication requirements between elements in the model. This model uses totally unsupervised training methods to store probabilistic representations of token sequential patterns in sentences, with no semantic extraction or analysis methods, yet its ability to complete sentences based on partial data with nearly correct grammar is arguably a 'cognitive like' feature.

#### *4.1.5.2 Overview of the confabulation model and sentence completion*

Over the course of this project we researched, implemented, and evaluated the performance of the confabulation model, focusing specifically for two example application problems that we call here sentence completion and intelligent on-line character recognition (OCR). In both of these applications the basic problem is to complete a partial natural language sentence in a plausible, sensible way, given that only a fragment of the input sentence is available, and given that the system has been trained by exposure to a large training corpus of textual electronic media (e.g. books and news feeds). Good solutions to the sentence completion problem could very well translate to other input modalities (i.e. audio and imagery), and map to solutions in several higher level application scenarios that are of interest to the military. For example, similar application include determining missing words in a garbled cell phone or radio audio transmission, multiple speaker disambiguation (the cocktail party problem), automatic textual annotation of surveillance motion imagery, prediction of sequences of events from video. Other immediate commercial or open-source uses include a conversational companion, and OCR for scanner and camera-based information extraction. A block diagram of the confabulation model for sentence completion is shown in Figure 11. It consists of a network of lexicon units that are interconnected by knowledge bases. The lexicon units correspond to word positions in a sentence, and each can express one of many possible words (or tokens) at its output, while the knowledge bases are data structures that capture the position dependent source/destination token co-occurrence pair probabilities of all tokens at all relative positions, summed and normalized across the training corpus. There are actually two levels of lexicon units, one a word level, where the outputs can be single words, and another a phrase level, where the outputs can be multi-word phrase tokens.

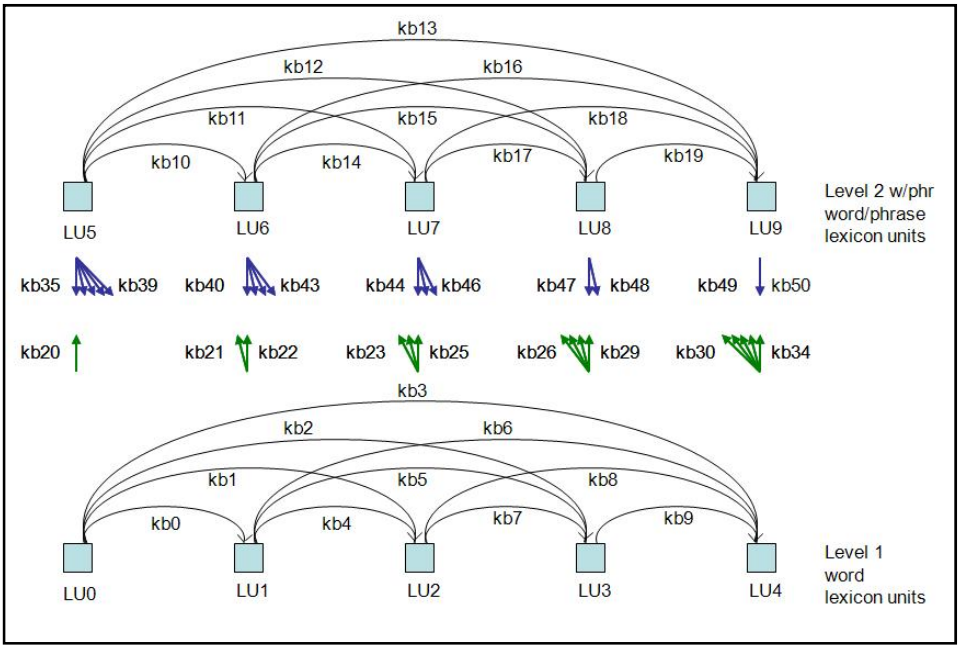



**Figure 11: Confabulation model network of lexicon units and knowledge bases.**

Written descriptions of the confabulation model in the literature to date are somewhat vague and certainly not explicit. Nevertheless, we have developed versions of the confabulation training and recall algorithms that we think embody the intended main features, and more importantly, seem to work as claimed by its proponents.

Variations of the confabulation model have been proposed that utilize two or more sentence units, and a third layer of lexicon units and knowledge bases above them, to operate on pairs or triplets of sentences. Other variations would utilize words extracted from audio, or objects recognized in images as the inputs instead of word tokens in sentences. Still other variations would involve multiple input levels operating on text, audio, and images, again with upper level lexicons and knowledge base connections to capture associations between these different input modalities. Although in the present project we have only studied single sentence configurations, we have proposed to study some of the extensions noted here under a follow-on project.

#### *4.1.5.3. Confabulation training and recall algorithms*

A detailed description of the exact training and recall algorithms we developed for the confabulation model is beyond the scope of this report, so in this section we only briefly and generally describe them in order to give some idea of their complexity and content.

Training the confabulation model generally involves using machine reading process to source a large number of single training sentences from a large corpus of electronic textual media (e.g. books and newsfeeds). For each sentence in the training corpus, we count the co-occurrences of specific source and destination tokens for every pair of token positions in the sentence. We also populate a global lexicon, or dictionary, with all of the unique tokens encountered. By “token”

we mean either a single word, or a pair of two adjacent words. The training process involves a first outer loop for training data file, a second nested loop for sentence in file, a third nested loop for starting token position in sentence, and a fourth nested loop for token destination lexicon unit. In the innermost nested loop, there are two token string lookups that find the indices of the source and destination tokens in the master dictionary and an increment of the count in the appropriate knowledge base. Because they appear inside four nested loops, deciding exactly how to store and lookup the lexicon tokens and knowledge base counts is critical to optimizing the speed and memory use of the application. We will show how fast and how large the size of these lexicons and knowledge bases grow with increased training for a number of possible choices for these methods later in section 4.1.5.4.

The confabulation recall operation involves a first step that transforms the accumulated counts contained in the knowledge bases to probabilities. This transformation includes normalizations to the sum of all possible output tokens in each lexicon, and to a minimum useful probability value. The confabulation operation begins with supplying the trained network with a partial sentence, and setting the outputs of those lexicons corresponding to known words in the partial sentence to their proper output values, or excitations. Then, the lexicons with unknown output states are traversed according to a particular order, and at each lexicon we calculate an output vector of activations that represents the probabilities that the lexicon output should be set to each possible token, then we perform a winner-take-all operation to decide which single token is excited at the lexicon's output. Each entry in the activation vector contains contributions from all of other lexicons that are connected though knowledge bases to the lexicon being calculated. This process also includes a thresholding feature that favors activation of tokens in proportion to the number of other lexicons that contribute to its activation, and a log is taken to compress the representation space. The recall operation is iterative, i.e. we record the intermediate output states of all of the lexicons after a first calculation, then we repeat the calculation of all unknown lexicon output states a second time (now with new lexicon output states supplied by the first confabulation participating), and the results are compared to the first calculation. If there are no changes in lexicon output activations, we are done, or if there have been changes, the lexicon output states are recorded again, and the calculation is repeated, until all lexicon outputs stabilize. The order of traversing the lexicons during the calculation can involve calculating at one new unknown lexicon to the right of known words on the lower word level, then calculating at unknown lexicons above and to the left on the upper phrase level (in a left-to-right fashion), repeating the forgoing until calculated lexicons on both levels stabilize, then calculating at one more lexicon to the right on the lower level, etc., until all lexicons on both levels stabilize. This process is referred to as "swirling" until convergence. Recall is graded by applying metrics that attempt to measure the plausibility or sensibility of the confabulated result. We discuss the metrics we used and the performance of the model using the metrics later in section 4.1.5.5.

Finally, we note that both speed and memory space constraints motivate us to parallelize both the training and recall operations. We want to be able to train from multiple data sources on multiple computing nodes, to train continually on new material, and to dynamically update trained networks with source material that becomes available only at a later time. However, the normalization process after training, that precedes recall, destroys the ability to retrain because the occurrence counts (that must be used to form column sums) are replaced by calculated probabilities. However, we can store the raw occurrence counts in trained network disk files before normalization, and do the normalization step only when a trained network is read in, before recall. This approach enables the merging of distributed training done in parallel, as well as additional subsequent training.

#### *4.1.5.4. Confabulation data structures, speed and memory*

The main data structures of the confabulation model are shown in Figure 12. They include the Global Lexicon, or dictionary; the Knowledge Bases; the Lexicon Units; and a KB Wiring array that specifies the network wiring (i.e. the lexicon unit indices that are connected to the input and output of each knowledge base). The size of the KB Wiring array is fixed and small (about 40*40*2 floats, or about 3.2K floats x 4 bytes/float = 12.8K bytes). The size of the other arrays is larger, and depends on how the data structures are implemented. We did a series of 4 implementations that included progressively more sophisticated data structures that we will call here (1) a full matrix version, (2) a linked list version, (3) an ordered linked list version, and (4) a hashed version.

Version 1 used full 3D arrays for the knowledge bases, and a simple large array for the global lexicon. It was very easy to program, and we used it to develop the basic training and recall algorithms. However, the knowledge base arrays are only sparsely populated, so this version was not useful for training on a large corpus because of high memory usage. Also, the global lexicon search method used in this version was a simple, but slow serial search.

Version 2 used linked lists to compress both the knowledge base arrays and the global lexicon array, a method that stores only the populated entries in the two arrays, saving memory. The search method for locating entries in the lexicon was again simple serial lookup, and the search method for locating entries in the knowledge bases was changed to serial search through the linked list. This version was used to develop the capability to merge the results of separate training sessions. It was memory efficient, but slow due to the serial search methods used to locate entries in the knowledge bases, something that was especially problematic during merging. To some degree, this was addressed by adding an array that noted the beginning and end indices in the linked list for each of the 800 knowledge bases.

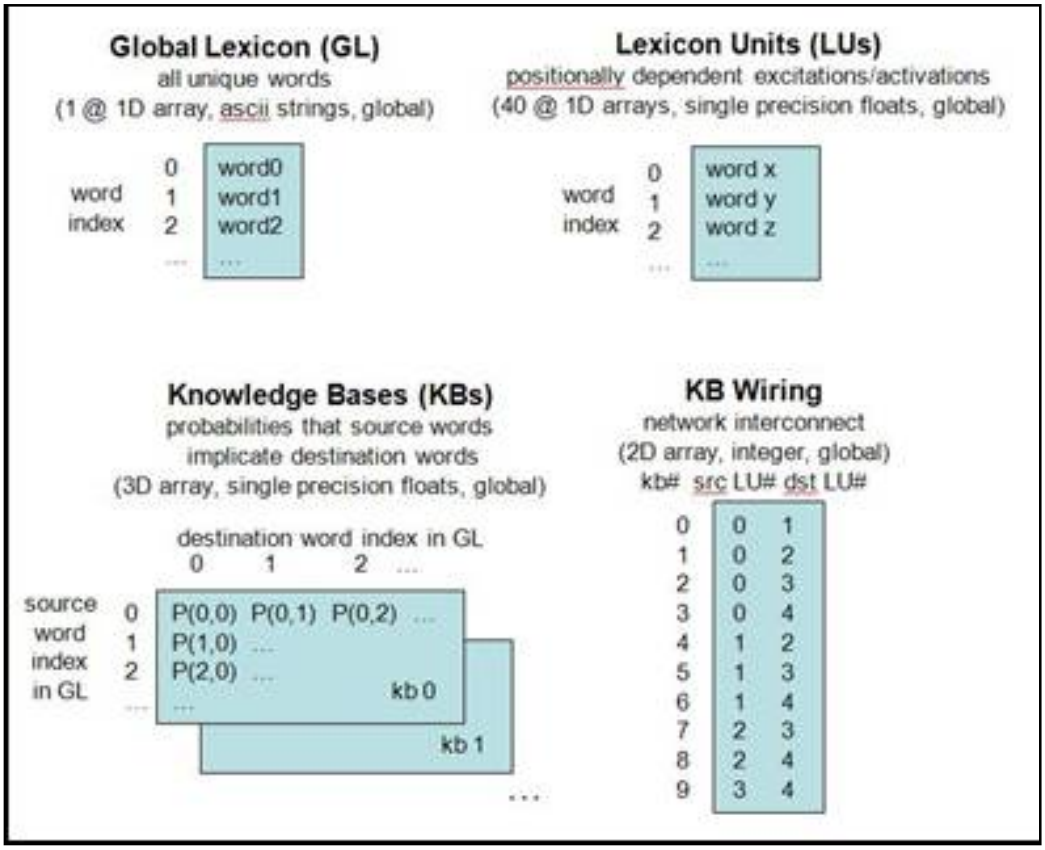



**Figure 12: Data structures of the confabulation model.**

Version 3 addressed the knowledge base serial search speed problem by ordering the entries in the knowledge base linked list according to source token ID and destination token ID. It also added a hash function/hash table method for storing entries in the global lexicon, as well as dynamic memory management for both the global lexicon and knowledge base linked list. This version was used to run large training and merging experiments on many books and newsfeed files, and to develop the performance metrics described in section 4.1.5.5. A distributed version was produced to train several books simultaneously on many cluster nodes. A variation of this version was also used to investigate architectural issues associated with hardware implementations of the confabulation training and recall algorithms (see section 4.1.5.6). A serious problem with this version was that during merges it held the data for all 800 knowledge bases for both files being merged in memory. When merging files containing tens of books of training, this exceeded the capacity of our largest available workstations (max 32GB RAM).

Version 4 added a hash function/hash table method for storing the entries in the knowledge bases, and a merge function modified to process each of the 800 knowledge bases one at a time. It also incorporated methods to handle ambiguities during sentence completions, i.e. to recursively produce all candidate completions in cases where one or more word positions had token choices that were equally probable. This version was about 200X faster than the previous versions, as shown in Figure 13, which enabled us to run large training and merging experiments to investigate the size growth of the lexicons and knowledge bases vs. the depth of training. We observed that knowledge base size growth was nearly linear vs. time during training, instead of power law. This version was also used as part of the BSB/confabulation hybrid model described in section 4.1.5.6. Finally, this version used a fully interconnected pattern of knowledge base connections between all lexicons, which is different than the 'feed-forward only' interconnection pattern shown previously in Figure 11. This new pattern allows known tokens in word positions to the right to influence the choice of unknown tokens in word positions to the left during completions, and enables new applications.

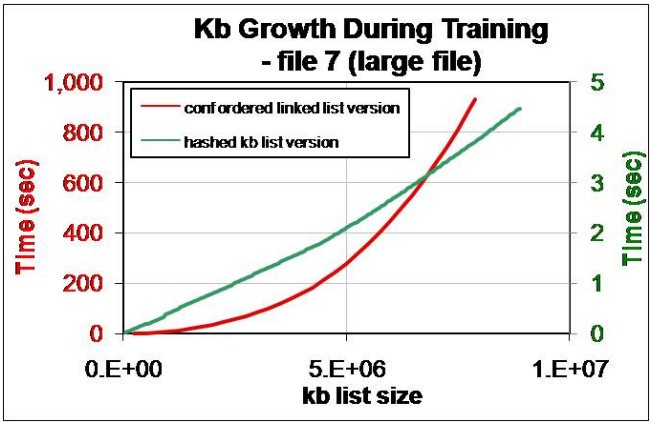



**Figure 13: Speed improvement of confabulation (training algorithm) version 4 over version 3.**

The total size of the data structures for the confabulation training and merge algorithms for version 4 are shown in Figure 14, for the case of training one book or newsfeed day, and with the assumption that the global lexicon is 1M tokens.

The sizes of the hash tables and hash collisions lists shown are sufficient for training very large books, and combining many tens of books of training, but could be adjusted to provide more capacity. For example, training on a large book (War and Peace) results in a global lexicon of 17,038 single word tokens and 145,015 two word tokens, which together required 1,781,503 bytes to store, and about 28M total knowledge base entries, and the largest of the 1560 knowledge bases had <40,000 entries in the hash collisions list. We also did experiments that trained 18 books and 23 daily newsfeed data files and merged the results, and the size of the resulting global lexicon was ~800,000.

**Data Structure Sizes**

- hashed knowledge base version 4
- training one book, 1M global lexicon tokens

| | | |
|---|---|---|
| Lexicon hash table | 2M * 2 ints * 4Bytes/int = | 16 MB |
| Lexicon hash collisions list | 0.5M * 2 ints * 4 Bytes/int = | 4 MB |
| Lexicon strings (1M lws) | 1M * 20 chars avg = | 20 MB |
| | | |
| Kb hash table | 1,560 * 32K * 2 ints * 4 Bytes/int = | 400 MB |
| Kb hash collisions list | 40,000 * 4 ints * 4 Bytes/int = | 0.64 MB |
| current kb sizes | 1,560 * 2 ints * 4 Bytes/int = | 0.01 MB |
| | | |
| Symbol list (1M lws) | 40 * 1M * 1 int * 4 Bytes/int = | 160 MB |
| | | |
| | | total 600 MB |

**Figure 14: Data structure sizes for version 4 confabulation training and merge algorithms.**

#### *4.1.5.5. Performance evaluation metrics and results*

The performance evaluation metrics and results discussed in this section fall into two categories: (1) speed and memory usage during training and merging; and (2) the accuracy and sensibility of completions made during recall.

The main speed metrics we used to evaluate performance during development of the training and merge algorithms were lexicon and knowledge base size growth vs. time (faster growth being better). An example of knowledge base growth vs. time was shown previously in Figure 13. Other metrics we used were total training time, total global lexicon size, and total knowledge base sizes, and the rate of change of these sizes as functions of input training data file sizes during sequential training. Figure 15 shows results for these metrics for experiments that trained 18 books and 24 daily newsfeed downloads separately, using both confabulation training algorithm versions 3 (red squares) and 4 (blue triangles).

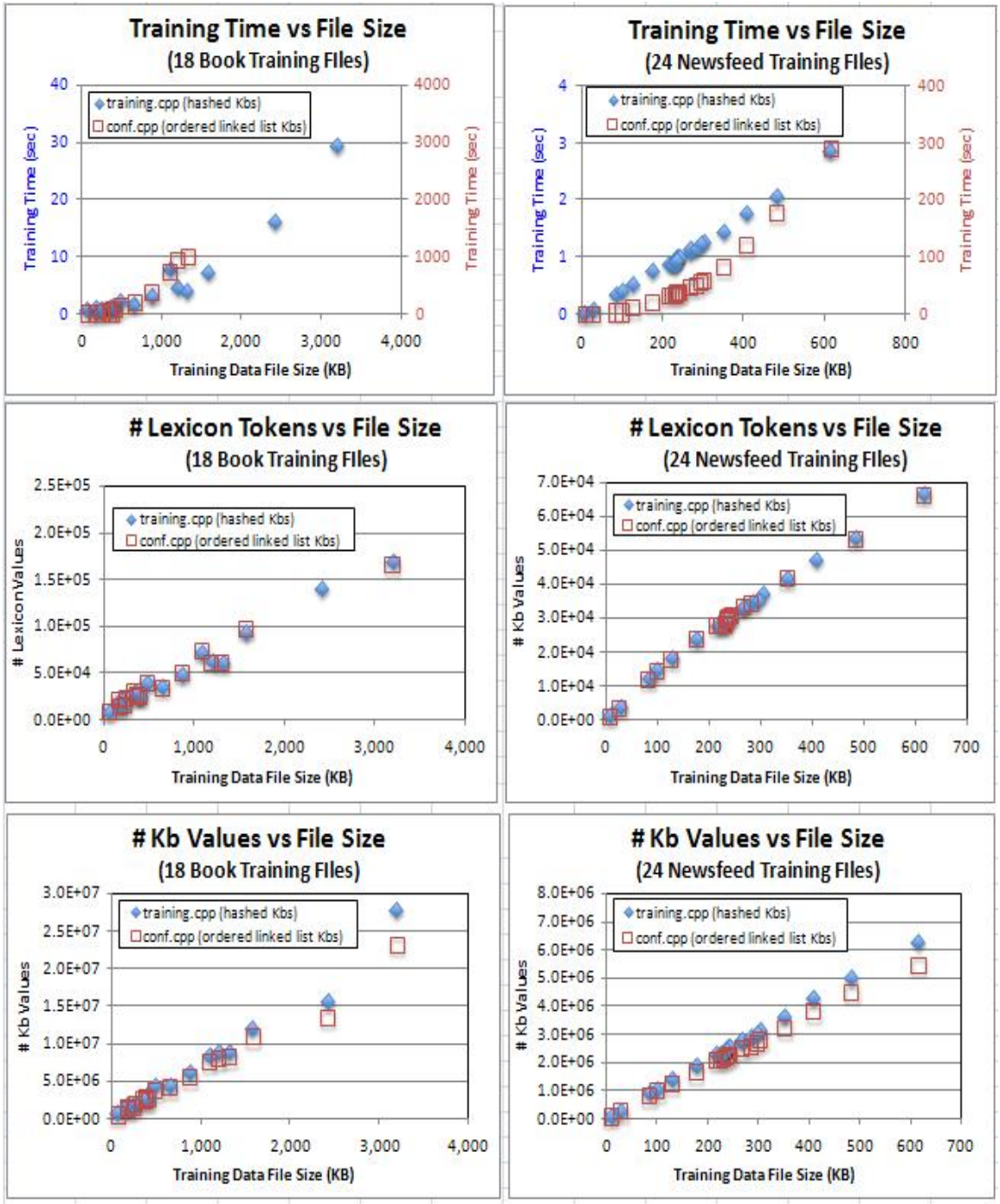



**Figure 15: KB size and training time for 18 books and 24 newsfeed files. for algorithm v3 (ordered linked list KBs) and v4 (hashed KBs).**

These plots clearly show that version 4 is about 100X faster than version 3, *total training time* is linearly proportional to training file size (# of bytes), and for version 4 both *total lexicon size* and *total knowledge base size* are approximately linearly related to input file size.

Another metric we evaluated for the confabulation merge algorithm was the merge time vs. the total # entries in knowledge bases of the second input file. An experiment was done that merged the previously trained 18 books, and the 24 newsfeed files, in a pair wise manner.

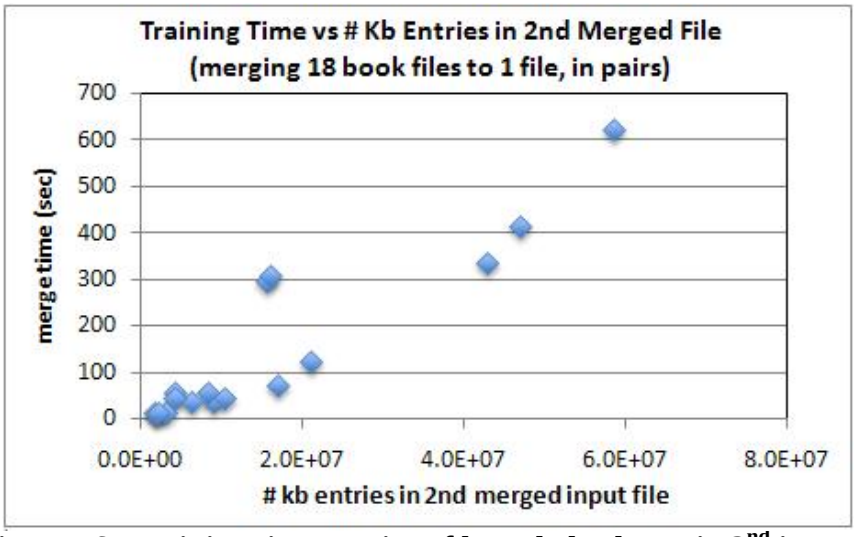



**Figure 16: Training time vs. size of knowledge bases in 2nd input file, merging 18 book files.**

Figures 16 and 17 show that merge time is approximately linearly related to the size of the knowledge base in the 2nd merged file.

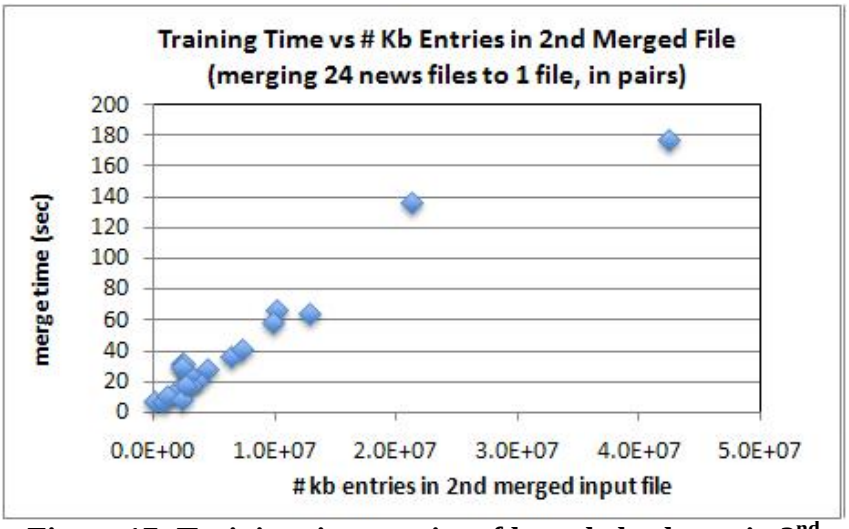



**Figure 17: Training time vs. size of knowledge bases in 2nd input file, merging 18 news files.**

We also looked at the portion of knowledge base and lexicon entries in the second merged file that were actually unique and added to the merged output file. It is expected that there will be some degree of commonality between the two input files, and that the overlap would be greater for greater training depth in the first input file. Figure 18 shows that for the case of merging several trained news files (points to the left), typically ~95% of the *knowledge base* entries in the 2nd input file are unique and are added to the output file, but for files that have captured more training (points to the right), only ~70% of the entries are unique and are added (points on right).

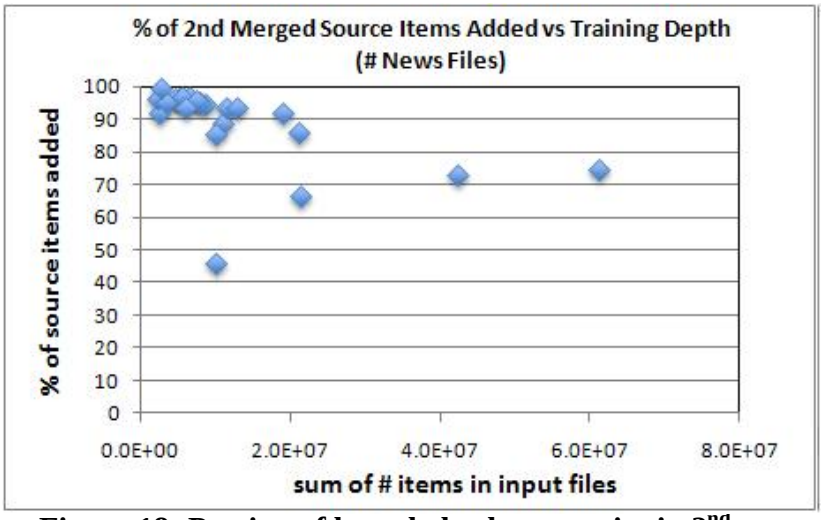



**Figure 18: Portion of knowledge base entries in 2nd merged input file added to output file.**

Figure 19 shows that as training progresses, only small portions of the entries in the 2nd input file's global *lexicon entries* are unique and added to the output file (10-20%, points on right).

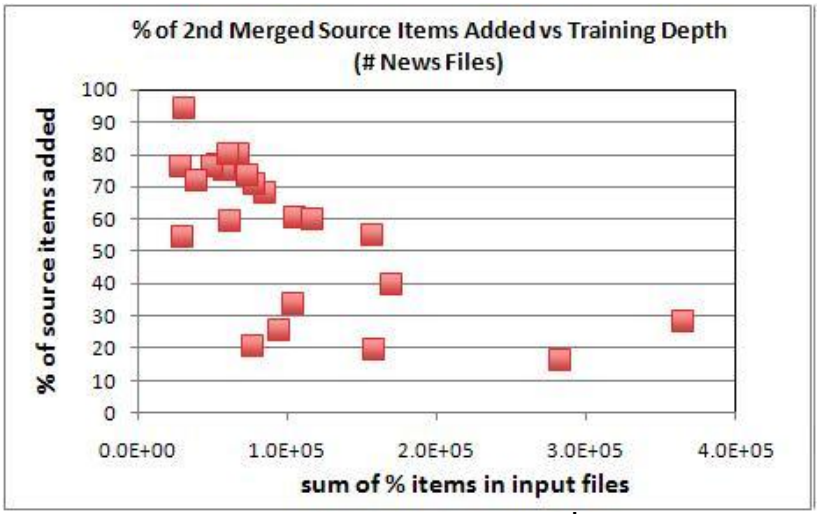



**Figure 19: Portion of lexicon entries in 2nd merged input file added to output file.**

Another way to observe this increasing overlap in both the knowledge base and lexicon entries is shown in Figures 20 and 21, where we plot the total # entries in the knowledge base and lexicon of the output file vs. the sum of the number of entries in the 2 input files. Both tend to flatten or plateau at the top, which is expected because there are only so many words in the language, and so many combinations of them as source/destination pairs that make sense.

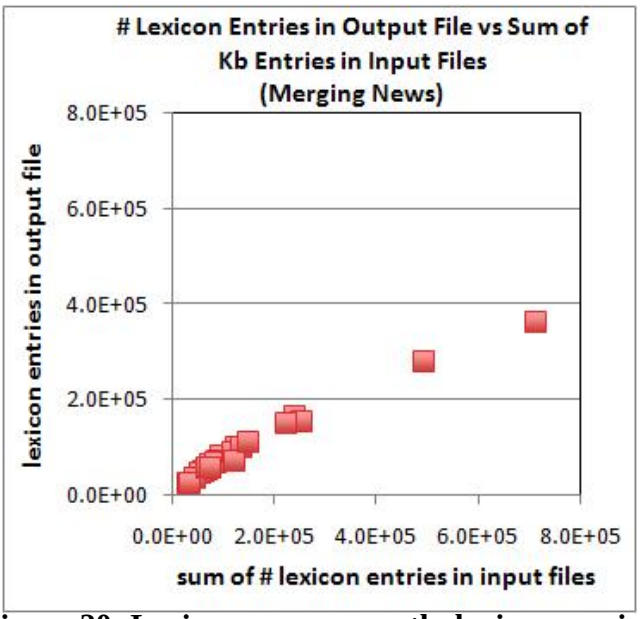



**Figure 20: Lexicon merge growth during merging of training on multiple files.**

The tests and metrics we used to grade the performance of the confabulation recall algorithm were straightforward, and automated. The first type of test (Test1) involved training the network on one or more input files, randomly drawing a number of original sentences from a particular training file, and randomly selecting a number of adjacent words (starting at the first word and continuing to the right) to include in a partial starter sentence. Both the original sentence and the starter sentence were stored in a file, the pairs were sequentially read, the starter sentence was completed by the recall operation, and the result was graded. To grade the completions, we counted the number of words in the original and starter sentences, calculated the difference as the number of words to be completed, and counted the number of words correctly completed by each recall. We noted the percent of words completed correctly for each sentence, and the total percent of words correctly completed over the set of test sentences. Then we produced frequency charts or histograms showing the number of sentences vs. the percent of words completed correctly. The first version of this metric (Metric 1) graded each completion only against the actual original sentence.

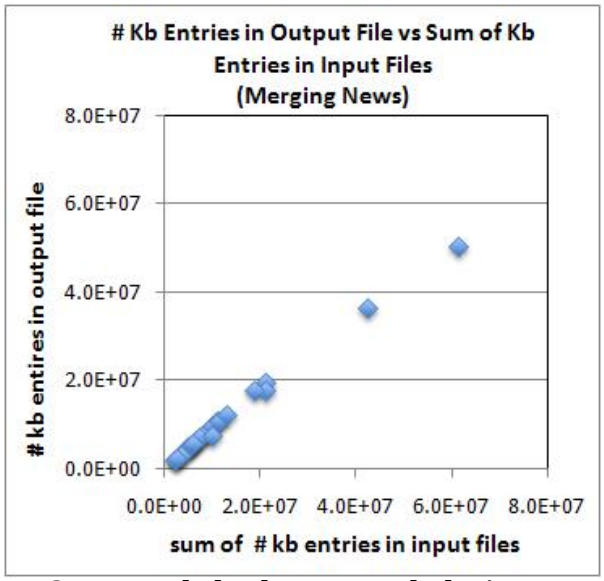



**Figure 21: Knowledge base growth during merging of training on multiple files.**

The second version of this metric (Metric 2) considered that the mapping from starter sentences back to original sentences was ambiguous, i.e. some starter sentences could have been derived from two or more different original sentences. Metric 2 involved searching the training data base to identify all possible original sentences, and giving credit for completions that resulted in any of the original sentences. Test 1was run on algorithm version 3.

Figure 22 shows typical results of running Test 1, graded with Metrics 1 and 2, for the case of a network trained on book 2, and tested with 100 starter sentences drawn from original sentences in the training data base.

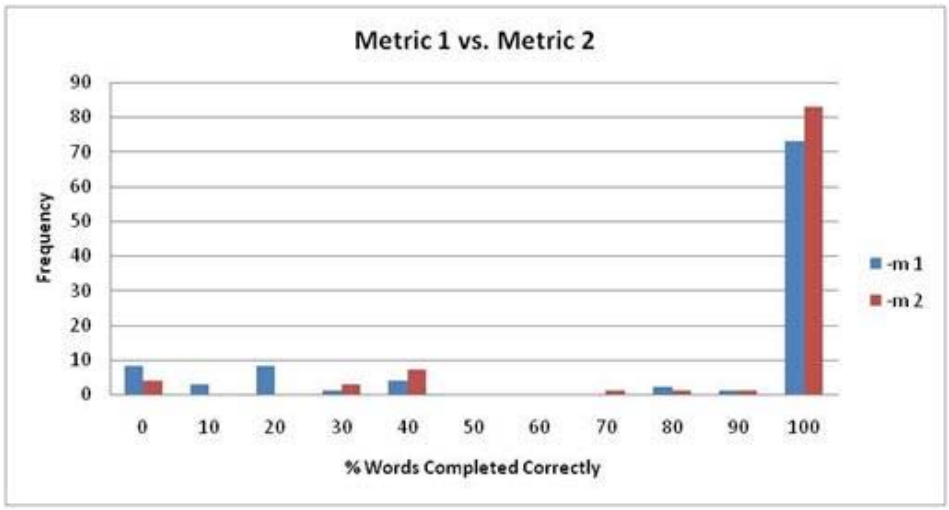



**Figure 22: Recall test grading with metrics 1& 2, 100 trained sentences, book 2.**

It shows that over 80% of the sentences were recalled with 100% accuracy, with small numbers of sentences recalled with less fidelity to the original sentence.

Figure 23 looks at the data from the same test in a different way, in terms of the percent of words completed correctly vs. the number of words in the starter sentences.

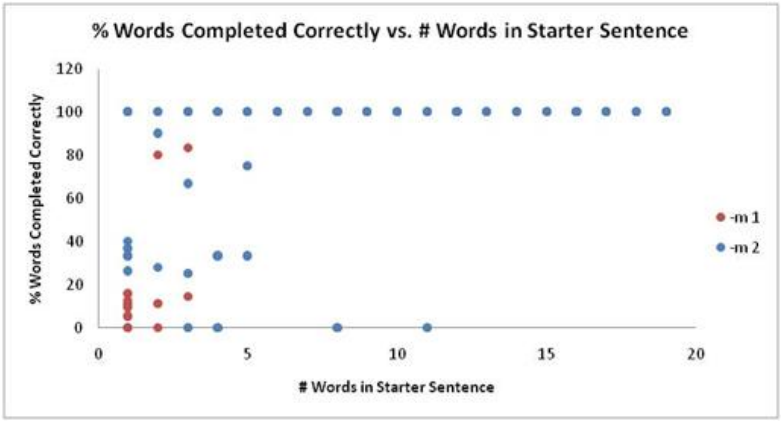



**Figure 23: Percent of total words completed correctly for the data of Figure 22.**

Clearly, sentences with fewer of the original words included in the starter sentences are less accurately completed. Also, as expected, Metric 2 gives higher scores to these short starter sentences, since some are ambiguous in terms of which original sentence generated them. We do not expect that confabulation recall should always complete starter sentences that exactly match an original sentence in the training base, because the algorithm uses probabilities to choose the later words that are completed. There can be cases where probability calculated for choosing a token in a word position is the same for more than one token. In such cases, the algorithm makes a list of the equally likely tokens, and chooses among them with uniform probability. This can give rise to completed sentences that are not in the training data base and indeed, this is the objective in building a confabulation system, i.e. to generate novel responses that may or may not be explicitly contained in the training corpus, but reflect the probabilities of token sequences present in the training corpus.

Figure 24 shows the results of running Test 1 graded with Metric 2 on several different networks trained on one book file, each with 100 sentences from the respective training files, in terms of the percent of total words completed correctly across all 100 sentences.

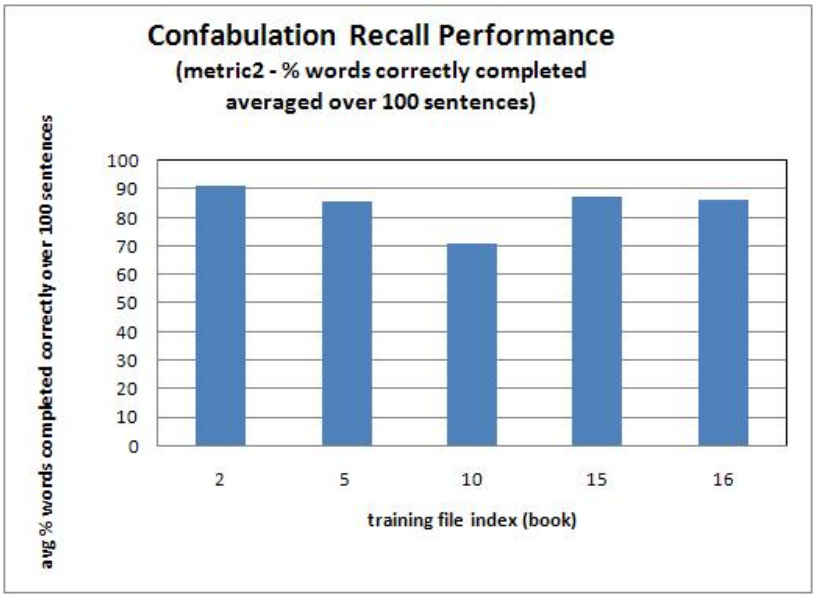



**Figure 24: Confabulation recall accuracy for algorithm version 4, test 2, metric 3, books.**

An additional issue that we studied was the effect of phrase token length on recall accuracy. We found that recall accuracy with phrase lengths 2, 3, and 4 were basically the same. Training with longer phrase lengths results in more knowledge base entries, so we typically used phrase length = 2. We also saw improvements in recall accuracy when a stop word list was used to prevent counting occurrences of a list of 25 of the most common words (such as ‘a’, ‘an’, ‘the’) as single word tokens. This prevented run on sentences of articles.

The second type of test for recall accuracy (Test 2) was run on the version 4 confabulation algorithm that used hashed data structures for the lexicon and knowledge bases, and fully interconnected lexicon units. This test simulated the type of errors that might be generated by transcribing noisy audio sources that captured partial conversations, or poorly transcribed text that might result from poor images of text pages captured by a scanner or camera. Test 2 involved randomly selecting a chosen number of original sentences from a designated training data file, selecting a percent of word corruptions to apply, and then randomly replacing that percent of the words in the original sentence with "?" to designate that the words were unknown. The unknown words in Test 2 were scattered throughout the sentences, rather than starting at a particular word and continuing to the right as they were in Test 1. Metric 3 was used to grade the results of Test 2. This metric counted the number of words to be completed (the unknown "?"s here), counted the number of words correctly completed, and calculated the percentages of words completed correctly for each sentence, and across all sentences, and then noted the percent of sentences that were completely correctly recalled. In this case it was not necessary to inspect the training files for the original sentences that might have given rise to ambiguous starter sentences (as in Metric 2 above), because in the version 4 algorithm, when there are multiple possibilities for token selections due to identical probabilities, each possible construction was made and tested. Recall accuracies were very good for this type of test and metric.

Figure 25 shows the results of running this test on several news files in turn. For each file, we completed 100 partial starter sentences derived from original sentences in the same file. The starter sentences were almost always perfectly completed by confabulation recall. Performance was better than for Test 1, probably because it is less likely that starter sentences are ambiguous using random word deletions than if the last few words are deleted.

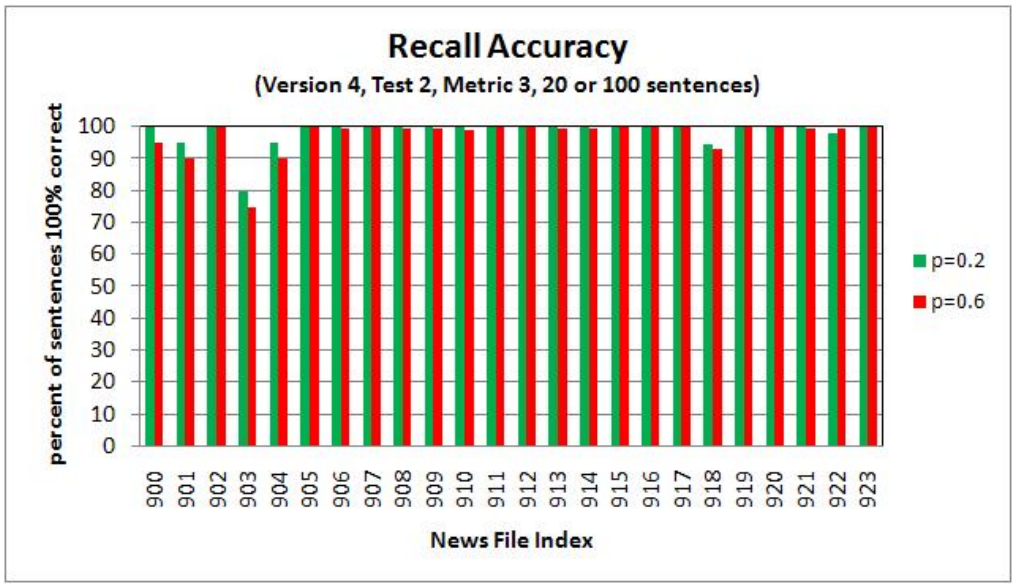



**Figure 25: Recall accuracy, 20% and 60% random word deletions, single news file training.**

We also did tests similar to Test2 described above on networks after merging training on multiple book and news files, both for the case of test sentences *in the training data*, and for test sentences that were *not in the training data*. Generally, the recall accuracy for starter sentences based on original sentences *in the training set* was not degraded by added training, as shown in Figure 26, where files moving to the right included more training.

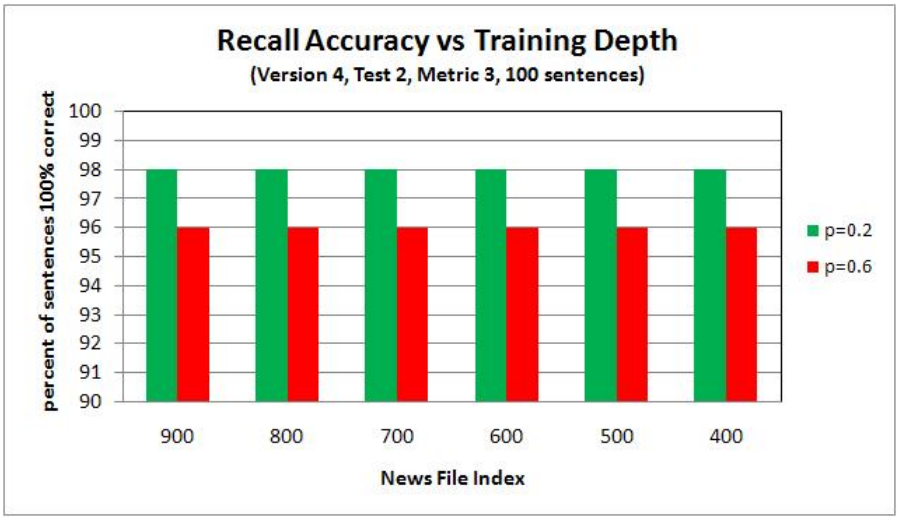



**Figure 26: Recall accuracy, 20% and 60% random word deletions, (deeper training on right).**

The tests of recall 'accuracy' for both the version 3 and 4 algorithms were also done for the case of starter sentences drawn from original sentences that were *not in the training data*. For example, we trained the version 3 algorithm model on several news feed files (900-923) merged the results, and tested completion using a set of starter sentences drawn from a *later* news feed file (937). We used Metric 1 to grade the result, since the 'original' and 'starter' sentences we used were not drawn from sentences in the training data base. In general the recalls did not complete the sentences according to the original sentences. This is not at all surprising, because that would amount to predicting the future. Although most of the completions did not make sense, some did, as shown in Table 8. It appears that some of the shorter 'starter' sentences derived from the 'original' sentences in the later file (937) were actually also present in sentences in the training data (900-923). Therefore, the observed sensible completions were actually identical to a previously trained sentence. This is an interesting demonstration of context based recall of previous training, triggered by a general query.

**Table 8: Example recall responses of confabulation version 3 algorithm, novel sentences.**

| |
|---|
| Starter ( 3) 'Some drivers have'<br>Output ( 20) 'Some drivers have been described as suspicious giving rise to fears that no action will be taken against cheats .'<br>Original ( 14) 'Some drivers have to wait more than a week to get into Russia .' |
| Starter ( 2) 'You can'<br>Output ( 20) 'You can t call what has happened here a real terrorist threat says Yahya Ould el Bara a university professor'<br>Original ( 17) 'You can never see something that hasn t been planned by the game designers he said .' |
| Starter ( 2) 'Suzanne Seggerman'<br>Output ( 20) 'Suzanne Seggerman the former maverick senator now parrot a broken GOP platform that he had spent his entire career voting'<br>Original ( 8) 'Suzanne Seggerman Games for Change http //gamesforchange .' |

A similar test was done using the version 4 recall algorithm, with a very short starter sentences that we expected were seen many times before. This kind of test mimics the way in which the confabulation model could be used in practice. Again we trained on multiple news files (24), merged the results, and then completed the purposefully general starter sentences. Table 9 shows that basically the algorithm recalled the many instances of sentences beginning with the short phrase that it had seen in training, similar to a content addressable memory. Remember that version 4 recursively does multiple completions based on using all equally likely choices of each unknown moving left to right. A significant difference using this recall method is that completions are much more likely to contain innovations not present in the training corpus.

**Table 9: Recall responses of version 4 algorithm for short common phrase.**

| |
|---|
| completed sen: John McCain Vietnam vet former prisoner of war is a Republican ` ` ` ` ` ` ` ` |
| completed sen: John McCain pledged to run a campaign that would be honorable ` ` ` ` ` ` ` ` |
| completed sen: John McCain despite all his claims of unique experience is just the wrong man to lead American foreign policy |
| completed sen: John McCain gets some room to maneuver out of the corner he had painted himself into on Iraq while |
| completed sen: John McCain holds a town hall meeting in Denver CO ` ` ` ` ` ` ` ` ` |
| completed sen: John McCain knows it ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` |
| completed sen: John McCain doesnt always tell us what we hope to hear ` ` ` ` ` ` ` ` |
| completed sen: John McCain no doubt attempting to summon the specter of John Kerry says the shift makes Obama unfit to |
| completed sen: John McCain should be ashamed of himself and immediately apologize ` ` ` ` ` ` ` ` ` |
| completed sen: John McCain does not have the potential to beat Barack Obama in a landslide and with all the negatives |
| completed sen: John McCain between the liberal and conservative approaches to solving our problems with no predetermination about who is correct |
| completed sen: John McCain speaks to the issues of women in the armed forces ` ` ` ` ` ` ` |
| completed sen: John McCain appears on CNNs American Morning in the am hour and then speaks at the League of United |
| completed sen: John McCain same old politics same failed policies ` ` ` ` ` ` ` ` ` ` |

We note that it is well known that it is still an open question to ask how much training will be necessary to enable any of the cognitive models being researched today to formulate sensible responses. In order to begin to answer that question for the confabulation algorithm, we believe that the model must be parallelized and deeply trained, much more so than the experiments we did here with tens of books and tens of daily news feeds, and we would like to do this as follow-on work to the present effort. We also think it will be very interesting to embed the kind of single sentence networks we focused on here in configurations that confabulate a whole response sentence that is triggered by one or more starter sentences. In fact, there is evidence in the literature that this is feasible and gives sensible completions.

Another remaining open question asks what is the best way to educate such a model. The training method for confabulation is unsupervised, but we need to think more about how to supervise the curriculum we use for unsupervised training. For example, what material do we want it to learn, and in the case of this model, what kind of sentence content and structure should be used to teach it? An interesting line of thought might be to consider how a confabulation network might be trained with information arranged in a special syntax, e.g. similar to what is developing in the domain of efficient content-based search of web pages.

#### *4.1.5.6 Prospects for speedup and scaling the confabulation model*

During this project, we also spent some time looking at ways to speed up and scale up confabulation training and recall. The algorithms are ideal candidates for parallel processing and their performance can be significantly improved with the help of application specific, massively parallel computing platforms. However, as the complexity and parallelism of the hardware increases, the design effort and implementation costs also increase. Architectures with different cost-performance tradeoffs were analyzed and compared in [51], which describes hardware designs that achieved ~1,000x speedup of the confabulation training algorithm, and ~3,000x speedup of the recall algorithm. Our analysis showed that although increasing the number of field programmable gate array (FPGA) processing elements (PEs) or the size of memories per processing element can increase performance, the hardware cost and performance improvements do not always exhibit linear relationships. Hardware configuration options must be carefully evaluated in order to achieve good cost performance tradeoffs. One interesting aspect of hardware performance optimization for confabulation recall is that the time (clock cycles) and memory space necessary to do a confabulation recall operation depend on the depth of training, i.e. how many entries are present and must be processed in the knowledge bases. So, some model of data structure growth vs. training depth is needed to estimate the expected maximum size in order to choose certain elements of the design. For example, in hardware it is possible to build content addressable memories that are appropriate for recalling the knowledge base entries. But there are speed/hardware resource trade-offs that can be made designing them, e.g. memory data word length (# bits). Longer words can be 'folded' to fit into narrow bit width memories to save resources, but at some cost in time. Also, statistical analysis of the number of entries on rows of various interconnecting knowledge bases shows that there is a distribution of row sizes. The most efficient hardware implementation might be different for different sized knowledge bases, so, a good design might include a variety of instances of evaluators that operate more efficiently on small or on large rows.

An example of a candidate hardware design for the confabulation recall operation that was developed during this work is shown at the top of Figure 27. The basic operations of the training and recall algorithms were deconstructed into elementary operations that could be implemented in hardware blocks, and they were analyzed in terms of whether they could be done in parallel, or in sequential pipelines that must block between stages.

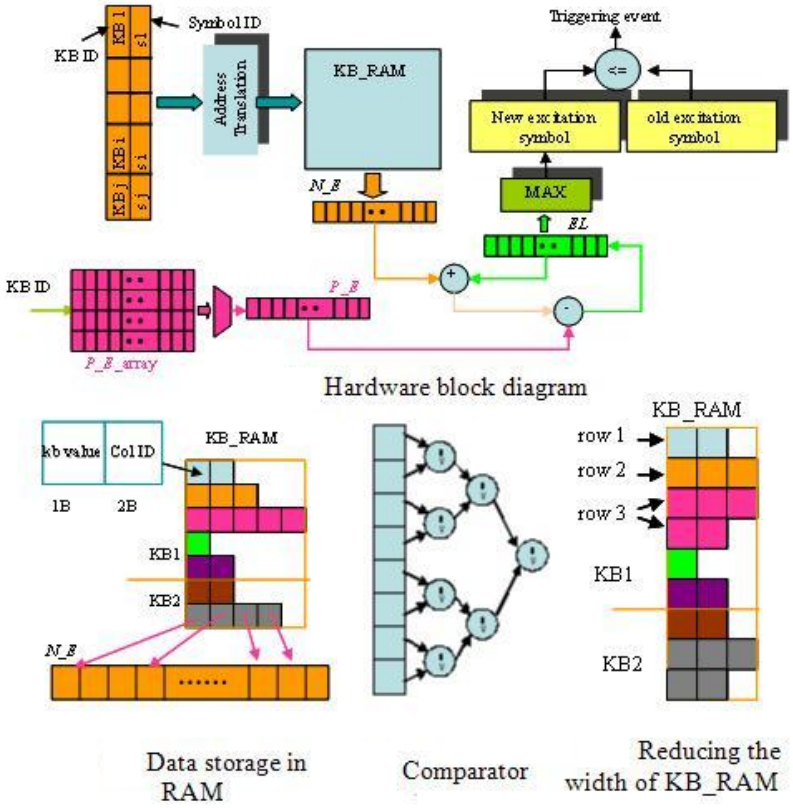



**Figure 27: Block diagram of hardware confabulation recall design.**

In a parallel hardware implementation, it is possible to multiplex the available hardware, i.e. to use a limited number of PEs to calculate a large number of different knowledge base calculations. Parameterized models were constructed and used to quantify speedup vs. cost tradeoffs cost for a number of design features, e.g. knowledge base memory width and number of PE's. Figure 28 shows an example of such a statistical model, and Figure 29 shows the result of using it to evaluate a trade-off between speedup and hardware resources (cost).

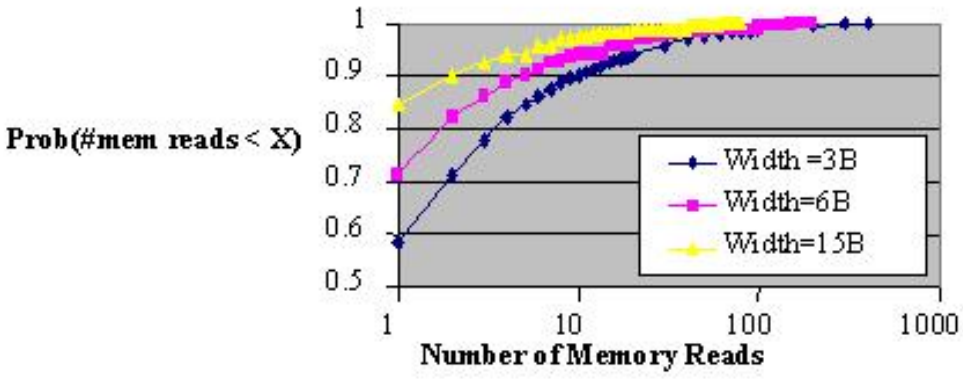
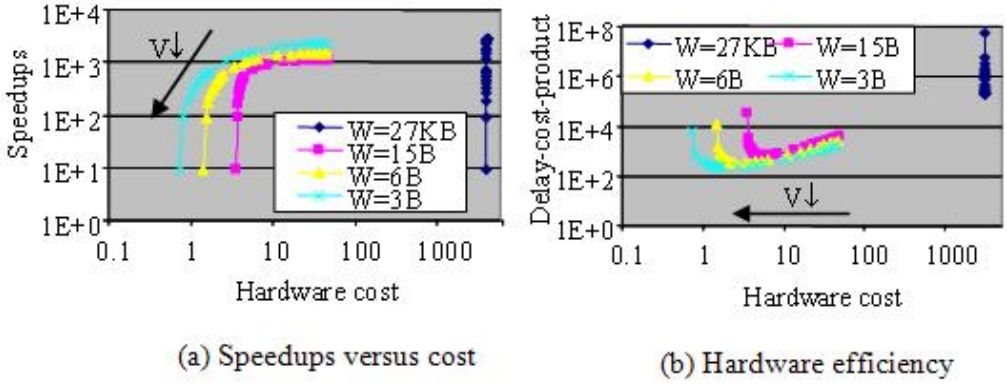



**Figure 28: Statistical model of a cost/performance tradeoff, FGPA confabulation recall.**

In this case, there are strong nonlinear relationships that influence the choice of memory width required to meet various speed and performance goals.




(a) Speedups versus cost

(b) Hardware efficiency

**Figure 29: Cost performance evaluation, confabulation recall FPGA version.**

The sentence level confabulation training algorithm was also ported to the CBE platform. A very simple parallelization strategy was use, i.e. the total number of knowledge bases was divided up across one or more SPEs for updating during training. An experiment was done to characterize the speedup performance of this version. It involved training a number of single books, each for the case of using 1-6 SPEs to process updates to the knowledge bases. The measured speedup averaged over all of the books was approximately linear vs. the number of SPEs used, e.g. reaching ~5.8X for the case of 6 SPEs. Interestingly, speedup scaling depended on the genre of the books, e.g. whether the book was scientific, classic, literary, children's stories, etc. This effect could be due to a concentration of knowledge base updates to a relatively small number of knowledge bases, rather than a wider pattern involving updates to a large number of knowledge bases. This would happen if the book's vocabulary is limited, or if the sentences have few words. This suggests that non-trivial parallelization strategies might prove useful for optimizing the performance of such models. For example, dynamic policies could be developed that consider the number of entries in a particular knowledge base, or a measured KB access rate. For example, we might want to map a heavily used knowledge base permanently to a particular SPE, and we might want to map lightly used, small KBs to a group that are swapped in and out of a common SPE.

Finally, we believe that both the confabulation training and recall algorithms will benefit from future parallelization on the PS3 based CBE cluster at AFRL. This future work is planned, and while it may hope achieve scalable linear speedup vs. # SPEs, the irregular memory address accesses that are an (intended) by-product of the hash table addressing schemes used in the sparse matrix storage methods used in these models may limit the peak SPU utilization factors to something less than what can be achieved for other problems that involve vectorized processing of full regular arrays (e.g. image processing, or BSB model processing).

### 4.1.6 A hybrid BSB/neuronal model

Based on our work in Task, two columnar based models were identified that could be used to develop large scale models. The first would combine BSB attractors and individual integrate and fire neurons into a minicolumn model. This approach would be challenging due to the computational load associated with simulating large numbers of neuron connections and their functional models. Such networks would tend to be sparsely interconnected, and would involve a multiply and add at each synapse for signal feed forward, and a second calculation (a compare and a multiply) at each synapse to account for input/output pulse timing dependent learning. Therefore, the "heavy load" is due to indexing necessitated by the interconnection sparsity, as well as the arithmetic of integrate and fire and spike time dependant learning at a synapses.

The second model combines BSB attractors with Confabulation. The interest is in the effectiveness of Confabulation as an expectation mechanism, to be used as higher cortical layer "gross modeling" while less abstract columnar mechanisms are developed between them and thalamic (sensor/motor) areas. Confabulation may also play a role in association of multiple sensory input data. This model is discussed further in the next section.

The exercises conducted using BSBs during Task 1 identified several issues. First, it is difficult to account for V1 orientation "pre-wiring" if the detection of angles is attributed to a biological attractor. That would mean the neurological equivalent of the BSB weight matrix, the synapses comprising the tight recurrent connectivity within a minicolumn, would all be pre-established. It is more likely the recurrent connections are somewhat random, and train based on experience.

Another issue is accounting for the use of simple cells within a minicolumn. Neuroscientific evidence suggests that non-recurrent "simple cells" do orientation detection based on the geometric patterns of their receptive fields receiving retinal ganglia signals. It has been confirmed in cats and monkeys that this geometric field does indeed come "pre-wired" [7, 22]. Therefore, there is no need for attractors to train-up basins of attraction specific to angles associated with a minicolumn.

There was no accounting in the literature for how a BSB network might deal with color and texture. Presumably, color and texture would be associated with a basin of attraction. A minicolumn model may be able to function that way; there are many minicolumns and each can represent a "primary" color, but intensity is part of color. A point attractor basin is a binary type of thing: you are in it or not. Larger BSBs representing greater cell populations, for example, a functional column, have less equivalent pixel density; they each represent tens or scores of retinal ganglia "pixels" with usually less than ten discrete attraction basins.

Contrast control (orientation contrast invariance) is yet another role attributed to minicolumns and not addressed in BSB network models. It is attributable to layer IV [35].

A feasible role of a small attractor within a minicolumn might be correlation of orientation evidence from afferent, lateral and expectation information. Each minicolumn is presumably excitable by the one angle associated with the orientation column it is in [25, 26]. Since these "BSBs" are small (32 neuron recurrence would be reasonable) they can have only a few reliable basins of attraction. It would be sufficient to have random wiring provide two or three such basins and be sharpened by early visual experience. Two basins can be used to recognize two opposite (antiphase) features; perhaps the same orientation in opposite direction (light to dark, and dark to light transitions). A third might be useful as inhibition: arrived at when it is clear no proper angle is in the minicolumn's field of view. The BSB input state vector would be a combination of complex cells detecting angle/ direction, laterals, and expectation data. The complex cells, a layer between the simple cells and the associative layer, would be assigned the role of translating afferent simple cell detections into state vector elements.

A useful BSB simplification falls out of this model: the weight matrices do not need to be unique. The BSBs are simply trying to "reach a conclusion." The patterns presented to it by the complex cell networks feeding it are presenting "evidence" that the feature associated with the minicolumn is present. The feature specific patterns are in the simple cells; signaling "present" or "not present" is all the BSB sees. This means one universal weight matrix can be used for all patterns. The model ends up with the following components:

- A 32 element BSB
- Two populations of simple cells; those corresponding to Parvocellular ganglia (Layer IVCb) and those corresponding to Magnocellular afferents (Layer IVCa).
- A set of complex cells for each of the ganglion afferent channels (Parvo and Magno), connecting them to the BSB.
- Complex cells connecting to lateral axons (incoming), connecting these to the BSB.
- Cells projecting the BSB state laterally, and forward.

No thalamic feedback was considered at this point, even though feedback along the optical radiations (geniculocalcarine tract) is four times the feed forward from the thalamus. It is assumed for now that features such as the cortical thalamic loop (loop of Archambault and Meyer) between the LGNs and V1, is not central to V1 efficacy. However, that loop is extensive, and may be revisited.

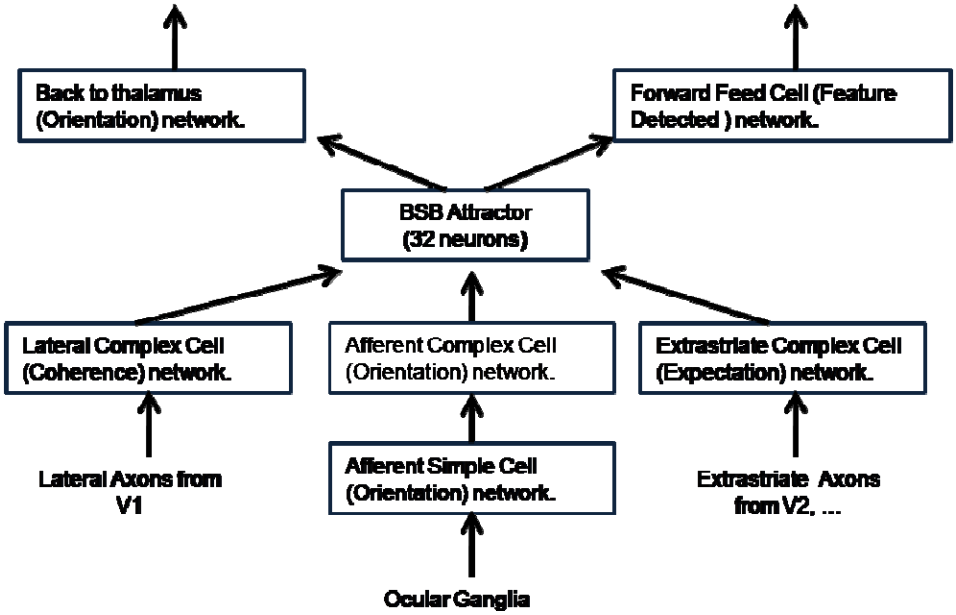



**Figure 30: Accounting for cells within a V1 minicolumn**

There are about 160 to 200 neurons in a V1 minicolumn, and the model tries to account for them in terms of selecting roles feasible in terms of current morphology information about minicolumns, and integrate and fire neuron model network characteristics. Figure 30 illustrates how the cells are assigned roles. The model makes use of complex cells to translate large numbers of inputs into small patterns that can be applied to the BSB state vector. The complex cells are likewise used to pull features from the state vector into (probably single axon) signals projecting out of the minicolumn: back to the thalamus, laterally, and upward to other neocortical areas.

The simple cells are assumed to be spatially tuned to one angle for excitation, and to its complement for inhibition. In each case, the simple cells are tuned to intensity transitions along an angle in either one of two directions: light to dark, or dark to light. Each simple cell is thought to be spatially tuned at one location in the field of view; field size and location vary, thus providing a level of invariance.

There are scores of simple cells in a minicolumn which need to be combined to produce a pattern indicating direction and strength of the "evidence" that the feature is present. That is the role assigned to complex cells between the simple cells and the BSB: to produce state vector inputs from the simple cell data. Likewise, complex cells between incoming lateral axons from other nearby V1 minicolumns and from extrastriate regions are assigned the role of translating those signals into a few BSB inputs.

The minicolumn also has "output drivers." These are shown in Figure 30 as arrows pointing out: feedback to the thalamus, laterally, and to feed forward brain regions. In each case, a translation can narrow down to a single axon output. The exact features put out are still speculative, but probably correlate (in V1) to orientation/direction, color and texture. Not illustrated in Figure 30 is provision for orientation contrast invariance, which may require complex cells modulating the simple cell afferent detectors. This aspect of the model will be addressed in future efforts. It is presently an active area of research internationally, and details of how the circuits work may well emerge from neuroscience labs in the near future.

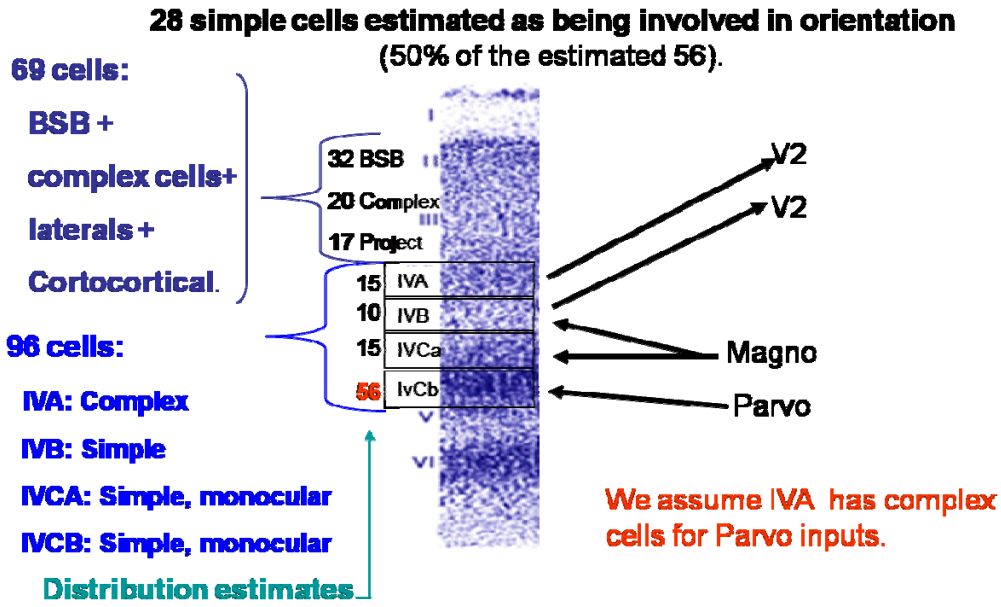



**Figure 31: Nissl stain densities and cell populations**

It may be noteworthy that the proposed small networks receiving the BSB state vector and outputting features may be functioning like read-out neurons in a Liquid State Machine. The BSB is a point attractor, and LSMs use randomly connected recurrent networks which may behave as point or limit cycle attractors.

The numbers of neurons in each block of Figure 30 may have significant effect on what the network can do. We try to estimate these numbers using available evidence from neuroscience. Figure 31 illustrates a classic nissl stain of a slice of V1 cortex, showing densities of the layers. The densities serve as a basis for a "first cut" cell population estimate in each layer and sub layer.

The initial estimate for the BSB attractor is 32 neurons. In part, this assignment is influenced by the technology behind implementing a BSB; it is efficient to use powers of two. The 32 element estimate is likely sufficient for how the model is currently envisioned to work, and may indeed have more than enough "space" in its state vector. The 32 element vector provides an estimated 4 well separated basins of attraction. The features which minicolumns in V1 are thought to detect are: orientation, direction, color/texture. It is not clear that all minicolumns within a

functional cell assembly in V1 engage in all four feature detections. The model can support "all doing all" by adding a connection in Figure 30 between thalamic input and extrastriate and lateral outputs, bypassing the BSB for color/texture. In that case, nonrecurrent neural networks may provide color/texture detection. However, there is neurological evidence that the minicolumns within the center of a functional column (CO BLOB radius) are orientation insensitive and color sensitive [40]. In that case, half of the minicolumns can be exclusively color/texture bound, and half orientation; the four basins can be dedicated according to their minicolumn role within a functional column.

The proposed model does not define how the non-recurrent networks are "wired," nor does it define cell populations within these. Instead, a baseline is being developed, and being configured for full scale representation of a V1 cortical area. It is being configured to be able to modify the estimates used to fill the "gaps" associated with the understanding of minicolumn architecture and function. Filling in the "details" is part of future investigation after the model is implemented. However, a "starting point" is defined which specifies:

- Numbers of simple cells associated with Parvo and Magno afferent fields.
- The field of view of minicolumn simple cells.
- The shape, size, angles, directions and spatial locations of all simple cell receptive fields.
- The weights of connections within the non-recurrent networks.
- The cell to cell connectivity of these networks.
- The lateral connectivity, both excitatory and inhibitory, using in-phase and antiphase connectivity.
- The translation of lateral information within a functional column assembly of minicolumns into a consensus vector representing orientation, direction, color and texture.

Color/texture detection within a minicolumn is not yet baselined.

It can be argued a columnar model like this, with distinct sub-networks assumed for hypothetical ocular functions, is clearly not consistent with the idea of universal minicolumn architecture throughout the brain. It appears to be too specialized. The rebuttal is that to some extent, each area of the neocortex is optimized to perform a specific job and that in V1 the circuits may form from the effects of pre-wiring, learning, and axonal routing due to stimuli. V1 in fact uniquely has a large cell population in layer IV. Other areas may "adapt up" local specialized circuits as well, starting from similar minicolumn architecture.

### 4.1.7 Hybrid BSB/confabulation model

One idea developed as a result of the investigation of the confabulation model was to regard it as a means to model prediction effects of "higher neocortex." For example, confabulation might be useful for modeling V2 while improving the V1 Model; V1 needs V2 because V2 provides predictive influences on perception. Likewise confabulation might later model medial-temporal lobes and later yet, frontal.

An experiment was designed to investigate how that might be implemented. In this experiment, BSBs were used to recognize individual characters in text strings. To make this challenging, many characters were deleted or smudged out, as shown in Figure 32. The confabulator was used as a mechanism to provide guidance to the BSBs in order to fill in missing information.

256 element BSBs were trained to recognize multiple character fonts using a 16X16 pixel array. The results of the "BSB initial pass" interpreting the text are passed on to a two layer confabulator: a word level and a phrase level. Characters which could not be recognized by the BSB in a limited number of iterations were passed to the word level as "unknown." Sentence length limited to was 20 words.

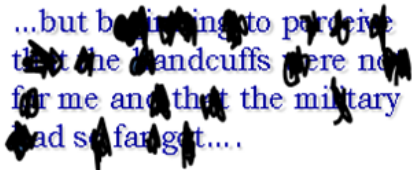


**Figure 32: Smudged text example**

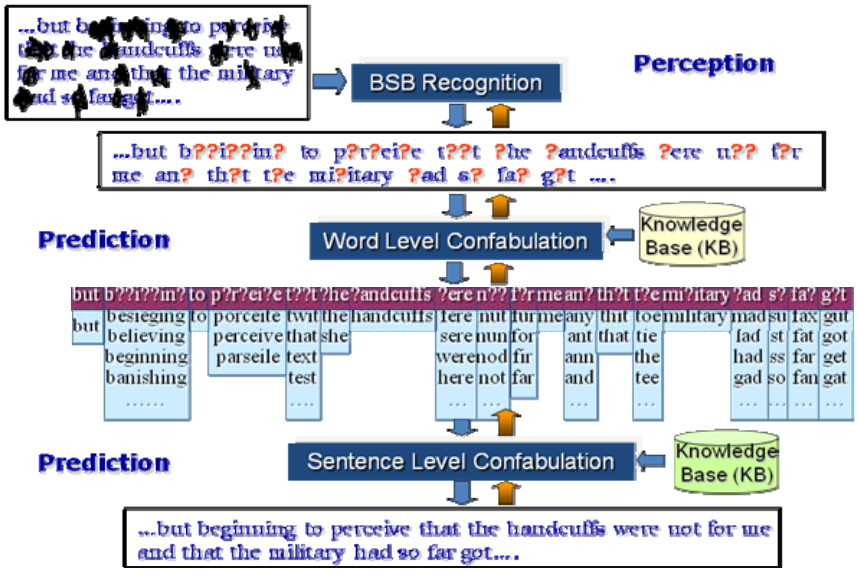



**Figure 33: The BSB/Confabulation Hybrid Model.**

The confabulation model (Figure 33) was trained using a list of 58,000 most common English words, and a set of 72 novels from classic literature, with an estimated total of 37 million words.

The confabulation layers accept tokens (letters) from the BSBs, apply word level and phrase level analysis, and then feed expectations back down to the BSB level. This feedback can be in the form of slightly increased excitations of pixels associated with characters of words supplied by the confabulation model recall algorithm. The BSBs then use the reassess interpretation of the pixels as characters, and this process can be iterated.

Recall performance using starter sentences drawn from trained sentences with 20% missing characters was almost perfect (~99% correct). Sentence recall using starter sentences not previously trained was in the range of 90% correct word selection and 60% correct sentence completion when 10% of the letters were missing; 24% and 86 % respectively when at 20% of letters were missing. Here by 'correct word selection' we mean that all unknown characters in a word were chosen correctly, and by 'correct sentence completion' we mean that all unknown words in the sentence were completed correctly.

Some examples of recalls that had high % correct word selection, but were counted as wrong at the sentence level are shown here it the reason for rejection highlighted in color.

Input: gra?ious ?o?dne?s ?r?c?o?s me what? ?one wit? th? pi?
Recall: gracious goodness gracious me whats gone with the pig
Original: gracious goodness gracious me whats gone with the pie

Input: i? was th? ?e?ge??t who had ?poken to m? and he was now look?n? round at the ?ompa?? with ??s
Recall: it was the sergeant who had spoken to me and he was now looking round at the company with was
Original: it was the sergeant who had spoken to me and he was now looking round at the company with his

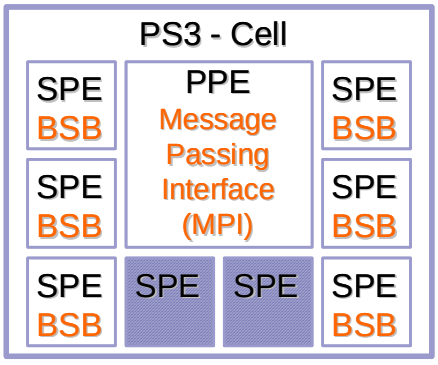


**Figure 34: Task distribution on one PS3.**

Details of implementing a 128-dimensional BSB model on the Cell processor can be found in [59, 65]. Referring to Figure 34, in the large-scale BSB model implementation, 128-dimentional BSB models are run on each of the six Synergistic Processing Elements (SPEs) on the Cell processor. The data communication functions are implemented on the PowerPC Processing Element (PPE), and the word and sentence level confabulation models are implemented on cluster head nodes associated with groups of CBE nodes. The BSB model was also implemented in an FPGA hardware version that achieved ~150 speedup over software [64, 65].

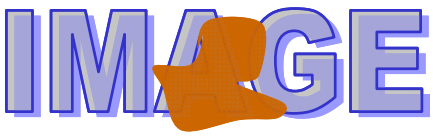
**Table 10: Performance, power, communication and modeling capabilities: 1 PS3 vs. 288 PS3s**

| | 1 PS3 | 288 PS3s |
|---|---|---|
| Peak computation performance achievable by BSB application: | 102 GFLOPS | 29.376 TFLOPS |
| Number of 128-dimensional BSB models supported: (10ms reaction time) | 3,000 | 864,000 |
| Equivalent mini-columns in the visual cortex of human brain: | 12,000 | 3,456,000* |
| Total Power Consumption: | 140 W | 40 KW |
| Achieved total network bandwidth for the communication test using MPI: | ~ 1 Gb/s | ~ 12 Gb/s |

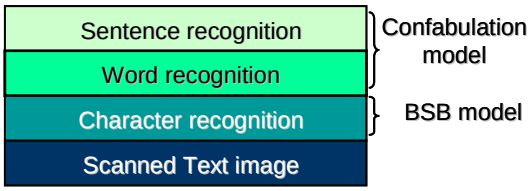
* The V1 layer of the visual cortex consists about 1,600,000 mini-columns.

Table 10 shows a comparison of the computing performance, communication performance, power consumption, and modeling capabilities between a single PS3 (1 CBE with 6 PSEs), and the whole cluster (288 PS3s). Theoretically, we can implement two V1 layers of the human visual cortex on this cluster.

Modern image processing software can perform image detection and pattern recognition with fairly high accuracy given the condition that the input image is clean and fully observable. Pattern recognition becomes extremely difficult, if not impossible, when the image is partially shaded or partially missing. For example, given the image in Figure 35a it would be difficult to recognize the middle character using only image processing techniques. However, this task is not difficult for a human as we fill in the missing information based on its context and our knowledge of sensible words that begin with IM and end with GE.




(a)




(b)

**Figure 35: (a) A partially shaded image (b) Layered architecture of intelligent text recognition.**

The intelligent text recognition system could potentially process scanned text images at very high speed, continuously learning from what has been read (excepting cases when uncertainty is detected), and can anticipate or predict not only the missing portion of words based character context within words, but also based on word context in other parts of the sentence. Such features may help the system to overcome OCR challenges that occur in the real world (or other sense-making problems in other domains).

This application was built using a hybrid of two cognitive computing models. They are the Brain-State-in-a-Box (BSB) Attractor model and Cogent Confabulation model. Cogent confabulation is an emerging theory proposed by Hecht-Nielsen. Based on the theory, the information processing of human cognition is carried out by thousands of separate thalamocortical modules that are collectively referred to as a lexicon or a feature attractor module. Different collections of neurons in the thalamocortical module represent different symbols. Knowledge is stored as the links between neurons and their strength. The cognitive information process consists of two steps: learning and recall. During the learning step, the knowledge links are established and strengthened, as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-take-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated, and the winning neurons activate other neurons through knowledge links.

The intelligent text recognition system can be divided into 4 layers as shown in Figure 35b. The bottom layer is the input of the scanned text image. The second layer consists of character recognition software based on BSB models. It tries to match the input image with stored images of characters in the English alphabet. The third and fourth layers are word and sentence recognition layers based on cogent confabulation models. They fill in the missing characters in a word and missing words in a sentence, respectively. The top three layers work cooperatively. The BSB layer passes the results of character recall information up to the word recognition layer, while the word recognition layer passes down word level context information that can be used by the BSB layer to perform pattern matching more efficiently. The word recognition layer sends words and partial words up to the sentence recognition layer while the sentence recognition layer also sends down sentence level context information that helps the word recognition layer to choose unknown characters more efficiently.

Figure 36 shows a simplified block diagram of the context based intelligent text recognition. The inputs at the bottom left are 2-word phrases, where a word is up to 20 characters. Each lexicon in the third layer (i.e. the word recognition layer) is associated with a character in the input word and each lexicon in the fourth layer (i.e. the sentence recognition layer) is associated with an input word. Therefore, there are 40 lexicons in the third layer and 2 lexicons in the fourth layer. In the rest of this section, we use the notation N-M to denote lexicon M in the Nth layer, with N

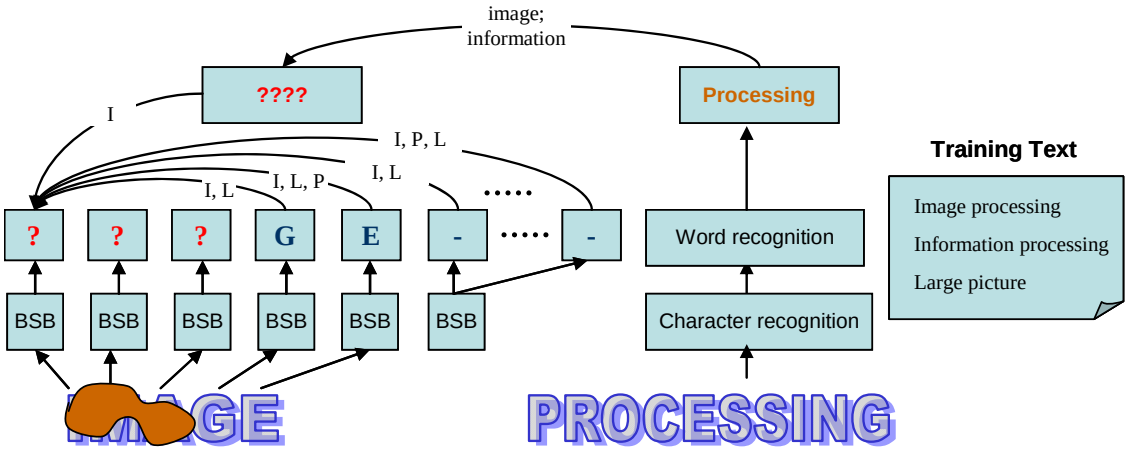



**Figure 36: An example of context based intelligent text recognition.**

as a Roman numeral. For example, III-4 indicates the lexicon 4 in the 3rd layer. The lexicons in the 3rd and 4th layers are fully interconnected with Knowledge links. For the sake of simplicity, we assume that the system has been trained with three 2-word phrases: "I", "information processing" and "large image". The input is the scanned image of 2-word phrase "image processing" with a smudge on the first 3 letters. Therefore the information sent from the second layer to the third layer is "???ge processing". Without context information, it is impossible to determine whether the first word is "large" or "image".

The word "???ge" is padded with 15 blank spaces and filled into the first 20 lexicons in the second layer. Since there is no missing character in the second word, the word "processing" goes directly into the lexicon IV-2. The letter "G", "E" and the word "processing" are considered as symbols that have already been activated. They excite missing symbols in the rest of the lexicons and the symbol with the highest excitation will be activated and further excite other missing symbols. For example, the symbol "G" in lexicon III-4 will excite symbols "I" (image) and "L" (large) in lexicon III-1 while the symbol "E" in lexicon III-5 will excite symbols "I" (image), "L" (large) and "P" (processing) in lexicon III-1. Overall, symbols "I" and "L" in lexicon III-1 will receive the same amount of excitation from the 3rd layer. On the 4th layer, the symbol "processing" in lexicon IV-2 will excite the symbols "image" and "information" in IV-1. Meanwhile, the symbols "G" in III-4 and "E" in III-5 will both excite the symbol "image" in lexicon IV-1. Overall, the symbol "image" receives more excitation than symbol "information" and will be activated in IV-1. It will further excite the symbol "I" in III-1 and eventually the symbol "I" will receive more excitation than "L" and be activated in III-1.

The actual system is more sophisticated than the simplified example. Besides lexicons for single letters and single words, it also has lexicons for each adjacent letter pair and adjacent word pairs in layer 3 and layer 4 respectively.

Stand alone software for each single layer in the text recognition system has been developed. In a randomly generated test where the number of missing characters in words ranges from 1 to 8 (or the percentage of missing characters of words ranges from 10% to 90%,), the stand along word recognition software completes the word correctly for 94% of time. The performance of word recognition degrades when the number of missing characters increases. Figure 37 shows that the percentage error of word recognition increases linearly as the number of missing characters increases. The sentence recognition software achieves more than 80% accuracy. When combined together, the accuracy of each component improves.

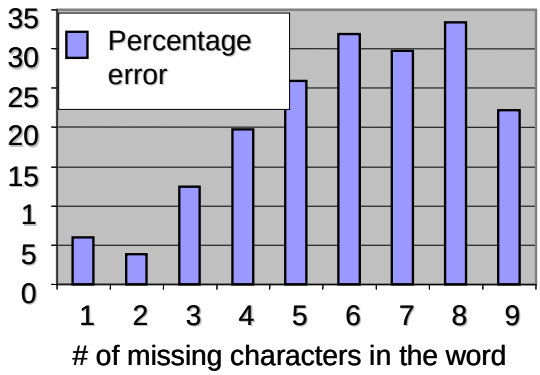



**Figure 37: Performance of word recognition layer for the hybrid BSB-confabulation model.**

## 4.2 Task 2: Evaluation of Large Scale Cortical Models

This task explored how models might be scaled up to a full scale brain model.

### 4.2.1 Hierarchical Bayesian model

The algorithm "scalability" (to full scale neocortex) was examined for the Hierarchical Temporal Memory model. Several sizes of the model were examined. The main model size examined had one level 3 node, 64 level 2 nodes, and 256 level 1 nodes. Level 1 nodes attach directly to afferents (16 pixels per node, 4096 total). The level 1 and 2 nodes are the computationally complex nodes, and were examined for "acceleration." The level 3 node does a simple arithmetic averaging function based on the outputs of all the level 2 nodes, and so was implemented in software.

The "neuromorphic equivalence" was estimated by the AFRL team to be roughly a patch of neocortex accepting 4096 "black and white" pixels. The estimate is based on a black and white pixel being modeled as opposing Parvocellular ganglion cells. There are about 800K "P ganglion cell axons" reaching the visual cortex from one eye. If considered in pairs (on-off center ganglion cell axons) then about 400K B&W pixels reach neocortex from each eye (800K total). One model's worth (4096) is roughly $1/200^{th}$ of a V1 primary visual cortex. Since V1 is roughly 1% of the neocortex (by area), it is reasonable to assume 20,000 of these Bayesian models would be approximately equivalent to a full neocortex (assuming cognitive efficacy held).

The algorithm was implemented on a Cray XD1 at NRL containing 864 2.0GHz AMD Opteron cores and 144 Virtex II Pro FPGAs. Two approaches were selected to measure scaling potential: parallel implementations with just AMD processors and parallel implementations with both AMD processors and FPGAs. In each case, time multiplexing is assumed; one algorithm instance performing the work of many of these models. We assume input data must "ripple up" the model, and ripple back down 5 times to stabilize a belief at the top. We also assume a "belief frame rate" of 5 beliefs per second needs to be sustained, corresponding to a saccade rate of 5 Hz. By this reckoning, the number of times a model needed to cycle per second to be "brain scale" is in the neighborhood of: $5 \times 5 \times 20{,}000$, or roughly 500,000 cycles per second.

The computational flow within each node was captured in a state machine for hardware implementation [38]. A VHDL implementation of the state machine was created for comparing performance (time) between a conventional processor and an FPGA implementation. On the FPGA, each level 2 node and its four level 1 children were grouped into a processing element. Up to eight such processing elements could be accommodated on a Virtex II Pro FPGA with 53,136 logic cells and a maximum clock speed of 190 MHz.

The nodes in this model make use of large training matrices. To optimize access to this data in the FPGA implementation, the data was stored in off-chip high speed SRAM memories. Each node in the model accesses its training matrix in a sequential manner to calculate its outputs. This sequential access allows the processing elements to stream in the matrix data from the off chip memory, thus reducing the on-chip memory requirements for each processing element. This also enables the processing elements to time multiplex the evaluation of a large number of nodes by streaming in the respective matrices.

The overall design of each node is shown in Figure 38. This figure shows only the four level 1 nodes in each processing element, and their associated on-chip training matrix buffers (there is a corresponding level 2 buffer and state machine in the processing element). Since data can be streamed in at a faster rate than it can be consumed, the processing element also includes a memory access unit to bring in data only when the on-chip buffers start being depleted of data.

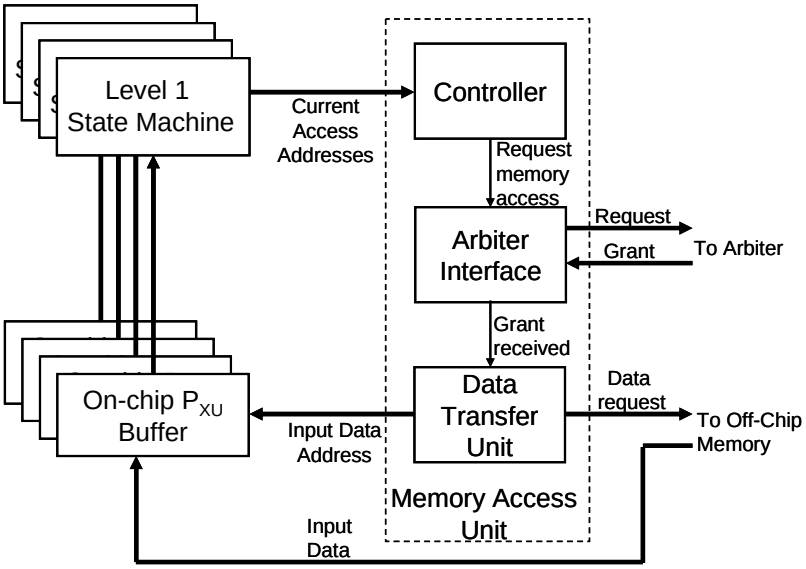



**Figure 38: The memory access unit in a PE.**

An arbiter was designed to allow several processing elements to share the off-chip data in a fair manner (as shown in Figure 39). The arbiter is required because nodes in the model can have varying run times because of differences in the training matrix sizes and may simultaneously request off-chip data. The nodes in the model were allocated to different processing elements based on this timing matrix size difference to evenly distribute the computations amongst all the processing elements on all the FPGAs.

The overall design with eight processing elements ran at a frequency of 123 MHz and consumed about 93% of the logic blocks and 86% of the block RAMs on the FPGA. This was excellent utilization of the FPGA resources. Of the 93% logic utilization, 2% was needed for the interface to the Rapid Array Processor connecting the FPGA to an AMD core. The optimized C implementation was run on a single AMD processor. An MPI version of the application was also generated where the master process evaluates the level 3 node, while the other processes evaluate the level 1 and 2 nodes.

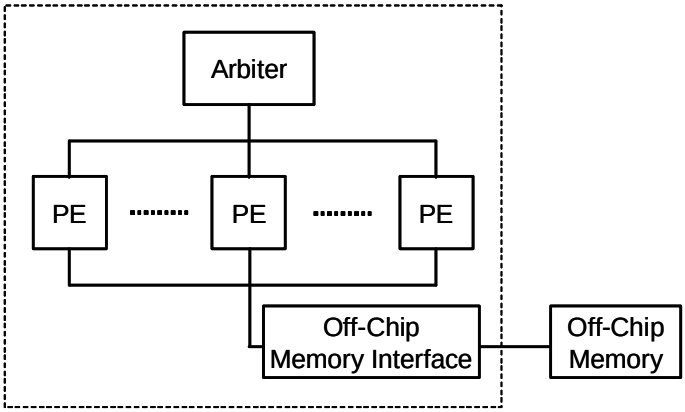



**Figure 39: A common arbiter controls access of the PEs to the off-chip memory.**

The 321 node network (256 level 1, 64 level 2, and one level 3) was implemented on two FPGAs and three AMD processors (two of the AMD processors were hosts to the two FPGAs). Two runs were carried out: one with the level 1 and 2 nodes running on the FPGAs, and another with the nodes running on the two AMD host processors. The third AMD processor combined level 2 beliefs generated from the first two processors to implement the level 3 node. When using the FPGAs, the two AMD host processors essentially performed MPI interface functions. The FPGA based system required 10.56 ms to evaluate an image through five passes of this network, amounting to a speedup of about 249 times over the software implementation.

This performance improvement comes from i) increased throughput due to the parallelism from the multiple PEs on each FPGA, ii) the multiple nodes evaluated in parallel per FPGA, and iii) the optimized implementation of the computations through hardware state machines. The smaller networks have simpler level 1 and 2 $P_{xu}$ training matrices. As a result, the fraction of time spent in serial tasks such as I/O (see Figure 40) is larger for these networks, thus leading to a lower improvement. These times do not include initialization operations such as opening, programming and closing the FPGAs, or initializing the Bayesian network with the training data.

The matrix operations listed in equations 1 to 6 for the Hierarchical Temporal Memory model are element by element operations (as opposed to dot products). Hence all the operations listed in equations 1 thorough 6 are multiply and divide operations, except for the comparison function in equations 3 and 4. By representing all the data variables in equations 1 to 6 as their logarithmic equivalents, the operations in the equations can be converted to additions and subtractions (the comparison operation in equations 3 and 4 is not affected). This simplifies the hardware implementation on the FPGA since adders are much smaller and faster than multipliers and dividers. The main data variables that need to be brought into each node are the input belief vectors and the $P_{xu}$ training matrix (used in equation 2).

$$\lambda_{product}[i] = \prod_{child} \lambda_{in}[child][i] \quad (1)$$

$$F_{xu}[j][k] = \pi_{in}[j] \times P_{xu}[j][k] \times \lambda_{product}[k] \quad (2)$$

$$m_{row}[j] = max(m_{row}[j], F_{xu}[j][k]) \quad (3)$$

$$m_{col}[k] = max(m_{col}[k], F_{xu}[j][k]) \quad (4)$$

$$\lambda_{out}[j] = \frac{m_{row}[j]}{\pi_{in}[j]} \quad (5)$$

$$\pi_{out}[child][k] = \frac{m_{col}[k]}{\lambda_{in}[child][k]} \quad (6)$$

The input belief vectors are of length 139, 1235, and 76 for the level 1, 2, and 3 nodes respectively. The $P_{xu}$ matrices were stored in compressed form and average about 5572 elements per node (after compression). All of these data variables are accessed in a sequential manner. As shown in Figure 40, the transferring of belief vectors between the SRAM and the FPGA for the level 1 and 2 requires about 4% of the total runtime. The transfer of the beliefs between the level 2 and level 3 nodes (between the FPGA and the AMD processor) required 12% of the runtime. We found later that this transfer time can actually be reduced significantly utilizing a DMA transfer between the FPGA and the processor, although this was not implemented for these results. The transfer of the $P_{xu}$ matrices from the SRAM can be completely overlapped with the level 1 and 2 node calculations. The overall computations for the level 1 and 2 nodes required about 78% of the runtime on the FPGA and 6% on the AMD processor for the level 3 node.

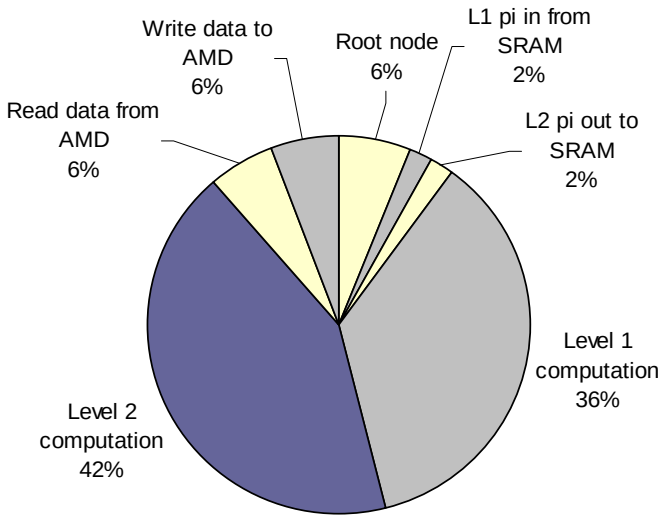



**Figure 40: Breakdown of runtime for one node in the FPGA based execution.**

Based on these results, the full-scale system would use roughly 1056 seconds to implement 1 second of real-time, on the 2 FPGA system (10.56ms/5 cycles × 500,000 cycles). Note that a full-scale system may have larger training matrices, thus increasing the time needed. To run at real-time, one would need roughly 3168 AMD processor nodes with 2112 FPGAs attached. The processing element based FPGA design can be configured to other algorithms by changing the state machine implemented in the processing elements.

Dr. Taha, at Clemson, is continuing to evaluate alternative platforms for Bayesian node computation. The Cell Broadband Engine is being investigated for the acceleration of the Hierarchical Temporal Memory model and Thomas Dean's hierarchical Bayesian inference model [8]. Figure 41 shows the overall structure of the Cell computing cluster at AFRL Rome Site. The cluster consists of 14 head-nodes that manage 14 sub-clusters. Each sub-cluster consists of 24 PlayStation® 3 (PS3) computers. Each PS3 has one IBM Cell Broadband Engine® processor as its CPU. The network speed among the head-nodes is above 20G bit-per-second (bps), and the Ethernet links among the PS3s is 1G bps. To achieve high performance on these systems, it is necessary to utilize vector operations and multithreading. The HTM model can be parallelized easily since the level 2 nodes do not have any connections to other level 2 nodes or to the children of other level 2 nodes. The Dean model on the other hand does have such connections, thus reducing its parallelism partially. However, for the small networks of Dean's model examined as preliminary work, about 4 to 6 of all the 6 processing cores on a PlayStation® 3 could be utilized. Larger versions of the model will have higher parallelism though since there will be more nodes in each level, while the number of levels will not grow significantly. Preliminary results indicate that the Cell BE platform can provide significant speedups over conventional general-purpose processors.

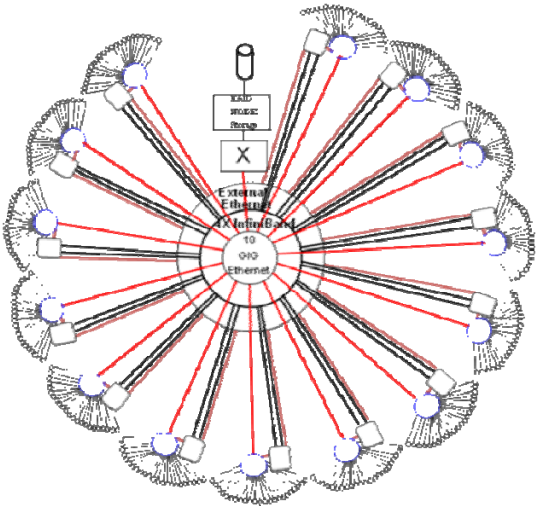


**Figure 41: Structure of the Cell computing cluster at AFRL/RITC.**

Dr. Taha is also starting to investigate the newer versions of these algorithms (both HTMs and Dean's model) in terms of their inference capability and acceleration. The newer versions of these algorithms capture the temporal domain in addition to the spatial domain (the versions of the models examined in this section consider only the spatial domain). With modeling of the temporal domain, the inference capability of the models is expected to improve significantly and follow processes in the neocortex more closely. Dr. Taha has also begun examination of accelerating spiking network models on the Cell BE [32].

### 4.2.2 Fixed point attractor network models

The Brain-State-in-a-Box (BSB) algorithm is a fixed point attractor. Other types of attractors include linear, strange and limit cycle. The term "fixed point" refers to attraction being directed at discrete points instead of other geometries. It has nothing to do with numeric types within the algorithm. Scaling and efficacy were examined in the context of large scale models. Scalability was approached by exploring how BSB algorithms can be accelerated by unconventional processing hardware. A conventional platform performance was compared to FPGA and Cell-BE implementations. Efficacy was examined by attempting to use BSBs to detect orientation lines as was described in section 4.1.3, except it was done using multiple orientation angles on a large scale V1 model run on a conventional platform.

The HHPC LINUX cluster at AFRL/RRS was selected as the conventional platform because of its availability and easy access, and because it provided up to 48 dual 2.2 GHz processor nodes with a 320 MB/sec myrinet interconnect. Each node is equipped with a 4 GByte SDRAM. A minicolumn was represented as a 26 element state vector BSB. Four were used for incorporating lateral communication. Six were assigned as expectation. Sixteen elements from the BSB were

dedicated to receiving black and white (B&W) pixels from an image. Each minicolumn had its BSB trained to distinguish the presence of an angle. Eight angles were defined. The image was preprocessed with a series (alternative) filters providing high frequency spatial filtering. The filtering emphasized contrast lines in the field of view, simulating Parvocellular occipital response.

The BSBs were assembled into groups of 64, representing V1 functional columns. There was one ganglion gray scale pixel, on the average, for each minicolumn. The 64 pixels associated with a functional column were averaged, 4 at a time, into 16 "windows." Each minicolumn viewed all the incoming ganglia through these 16 windows. Each minicolumn BSB trained for a different pattern. Each angle had equal representation in a functional column.

The functional columns were gathered into assemblies of 64, so that 4096 BSBs were contained within a single LINUX process. An image was distributed retinotopically over identical 196 processes representing a complete V1 hemisphere. Only 96 nodes were available, so only 96 could actually be instantiated for timing tests. Each process was independent of the others, but lateral messages were used to communicate minicolumn state vectors. A "Pub/Sub" communication system was used to provide a convenient way to scale a test from one to 196 processes (multiple processes per node in that case).

BSB states were each mapped into a small (4 element) "tag" vectors to represent the state; these vectors were the contents of the lateral messages. Each possible angle was assigned a tag pattern; the magnitude of a tag was an inverse measure of distance from a basin. For any orientation column in a functional column, the strongest tag magnitude was used to select which incoming lateral communication to use and distribute into the state vectors of all minicolumns in the orientation column. Laterals received were used to override state vectors (rewrite the portion of the state vector corresponding to lateral information). Any BSB cycle uses the lateral, local and expectation data to produce a next state because all three information sources are incorporated as fields into the state vector.

One "efficacy test" intent of this configuration was to examine how well orientation angles could be distinguished by using BSB attractors to examine square receptive fields. The idea was to absorb the simple cell efficacy into the BSBs, having each BSB using a weight matrix customized for its specific orientation. This approach keeps the connectivity between the retinal ganglia and minicolumns amenable to using continuous vectors, side stepping the problem of sparse connectivity associated with modeling simple cells at the cost of using individualized weight vectors.

Another intent was to examine the ability of the Pub/Sub communication model to distribute visual data pieces over a large set of processes. For this, all nodes were utilized, and then 196 processes were mapped onto the 96 actual processors. The selection of the lateral part of the model was ad-hoc. No model of it had yet been found in the literature search; the component of the literature search which might fill in some information on lateral connectivity had not yet been conducted. A simple nearest neighbor (functional columns level) winner take all method was applied.

The results of orientation efficacy testing were that orientation lines of simple grating patterns could be identified reliably for horizontal and vertical patterns, and 45 degree angles seemed to be detected well too. Other angles between these were unreliably detected, often confused with horizontal, vertical and 45 degrees. The relatively small field of view associated with these detections (a square of 16 pixels) limits the ability to distinguish small angle variations. Larger fields of view, using a BSB to detect angular features, require larger state vectors. There is a tradeoff to be made in the use of attractors to do the work of feature detection performed by the simple/complex cell networks: complexity increases with the square of the size of a BSB attractor, while it rises linearly with the number of cells in a simple cell type feature detector.

The few "runs" made to explore efficacy of the "lateral model" did not reveal any increased success or even a tendency to detect illusional data, though one such result did illustrate a faster convergence when expectation data was applied. The lateral exercise did increase confidence in the "pub/sub" message model; no waiting on the pub/sub server was noticeable.

The 6 state vector elements dedicated to expectation were kept neutral at the beginning of each perception trial. Expectation did not contribute to feature recognition; its neutrality was modeled by a zero expectation. The existence of the field affected the speed of resolution of the BSBs (slowed it down in this case) and therefore improved the neuromorphic realism. The pub/sub messaging did not apparently slow down when all HHPC nodes were used in a 96 process experiment. The experiment was able to manage about $^1/_5$ real-time. Performance was near enough to "adequate" that resources were not put into tuning. This is especially relevant since one node was burdened with additional duties – it managed the image preprocessing, fragmentation into small fields, and distribution.

Conclusions drawn from this "quick look" exercise:

- The pub/sub messaging model provides a very flexible method of system configuration without having to attend to details of physical node availability and node inclusion or exclusion. The system middleware used for this, a version of JBI developed at AFRL/RI, performed well within efficiency needs.

- It is possible to execute 4096 small BSB functions within a process, on 2.2 GHz platforms, at reasonable rates (1/5 real-time represents executing each of the 4096 functions 5 times/second, in context with the overhead of messaging, and a lateral model.)

- More work is needed to understand how minicolumns interconnect laterally.

- The afferent ganglia connection details are not well understood (in neuroscience literature).

- The BSB training, representing "a priori neuro-wiring" is not likely to be representative of natural phenomena. Each minicolumn BSB in this model had a hand crafted learning pattern for an orientation line. No evidence exists for this pre-trained recurrency in the V1 of a brain, but evidence exists for orientation line discrimination at birth [22]. There is strong evidence of geometrically arranged (simple cell) receptive fields; these simple cells are apparently not in a recurrent path local to a minicolumn. This is an argument against V1 using local recurrence to detect angles.

The use of Field Programmable Gate Array (FPGA) technology to accelerate BSB speed was investigated. The objective is to be able to support a very large number of minicolumn or functional column models. A minicolumn BSB is considered here to be a 32 element state vector BSB. A functional column BSB is considered to have 128 elements in its state vector. There are an estimated $10^8$ minicolumns in a neocortex, and about $10^6$ functional columns.

WILDSTAR II PCI cards with dual Virtex II XC2V 6000 FPGAs were used to develop BSB recall functions. Six 2MB local memories are associated with each FPGA, providing 5.5 GByte/Sec bandwidth. The BSB implementation on these units used fixed point arithmetic instead of floating point. A 32 element recall ran in 32 clock ticks (100MHz), producing recall in 320 ns. This worked out to be a 40X speedup compared to a 2.4 GHz PC platform conventional software implementation. Recall for a 128 element BSB is estimated at 160X speedup (204.8 microseconds on the PC platform), 1280 ns. These timings did not take into consideration the input and output time needed for the state vectors, and weight matrix loading. Note however that test cases described in Task 1 indicated the BSB algorithm needs to be cycled about 5 times for convergence with a 32 element vector, and 10 or more for 128 elements.

State vector I/O is relatively light duty. For each recall, a 32 or 128 element vector is loaded onto the FPGA, and then loaded off. Each element is a 16 bit fixed point value, so net data movement is 64 or 256 bytes, depending on state vector size. Timing at 5.5 GBytes/Sec translates to about 12ns for 32 elements and 48 ns for the 128 element case.

Weight matrix I/O requires a weight vector to be uploaded into an FPGA prior to a recall. No such upload is needed for the 32 element BSB in certain cases (See the discussion on the hybrid BSB/Neuron minicolumn in the Task1 discussion). The matrix size is 32X32 and 128X128 for the 32 and 128 element BSBs, respectively. Upload time is likewise: 186 ns and 2,979 ns.

FPGA I/O time and processing time can be overlapped and therefore restrict total time to the time needed by whichever is greatest: processing or I/O. We estimate that cognition efficacy requires a BSB to perform a full recall on new data roughly 10 times per second. Estimates for minicolumns/neocortex and functional columns/neocortex are $10^8$ and $10^6$, respectfully.

Scaling differences between minicolumn and functional column modeling is dramatic, but does not account for connectivity between the BSBs.

**Table 11: Timing estimates for FPGA speeds including multi-cycle recall and IO**

| State Vector Size | Recursion cycles/ recall | Unique Weight Matrix | IO Time in ns | FPGA execution time in ns, includes all cycles. | PC execution in microsecs | Peak FPGA full recalls/sec | Estimated recalls needed per sec, full scale: | Total FPGAs needed, full scale |
|---|---|---|---|---|---|---|---|---|
| 32 | 5 | Yes | 358 | 1,600 | 64 | $6.25X10^5$ | $10^9$ | 1,600 |
| 32 | 5 | No | 172 | 1,600 | 64 | $6.25X10^5$ | $10^9$ | 1,600 |
| 128 | 10 | Yes | 2,979 | 12,800 | 2048 | $7.81X10^4$ | $10^7$ | 128 |
| 128 | 15 | Yes | 2,979 | 19,200 | 3072 | $5.20X10^4$ | $10^7$ | 193 |
| 128 | 20 | Yes | 2,979 | 25,600 | 4096 | $3.96X10^4$ | $10^7$ | 253 |

Not shown in the table is a summary of the total "basin space". It is an interesting way to compare how larger and smaller attractors scale. A 128 element BSB can support 19 basins. This can be regarded as a BSB taking on one of 19 possible values. Likewise, a 32 element BSB can support 4 basins. A functional column scale brain model has one million BSBS, each with 19 basins. Thus one million of these BSBs has a basin space of $19^{1,000,000}$. The minicolumn model likewise has a space of $4^{100,000,000}$, roughly $4^{99,000,000}$ times the size of the functional column space.

BSB Performance acceleration based on CELL-BE technology. The use of IBM Cell-BE technology (Sony PlayStation® 3 platform) to accelerate BSB performance was investigated. Runtime measurements show that we have been able to achieve about 70% of the theoretical peak performance of the processor when implementing a 128 element vector using a matrix shuffle strategy to improve Cell-BE SPE instruction utilization [59]. (43% Peak was later achieved on 32 element vectors within the model described in section 4.2.5: Hybrid minicolumn: BSB + Neurons Acceleration.)

The 128 element BSB recall algorithm was implemented on a single SPE element of the Cell-BE architecture. The complexity is 33,280 FLOPs/ recursive cycle. Ten cycles are needed for convergence yielding 332,800 FLOPs/ recall. Peak efficiency corresponds to all floating operations being performed as quad word operations, with all other (non-floating point) instructions executing in the parallel instruction pipe. In this case, peak is 332,800/4 = 83,200 Quad Floating ticks. Each recall needs a weight vector load, a state vector load and a state vector unload (66,560 bytes) equivalent to 4160 quad word transfers (one quad word per tick). Compute to DMA peak ratio is therefore 83200/4160 = 20. Double buffering was used to overlap data transfer of weight matrices and state vectors with processing. Six BSBs can be run in parallel on a PS3 version of the platform. Efficient implementation on an SPE requires careful attention to aligning data for maximum effectiveness of intrinsic functions. Loop unrolling is essential as well to maintain the dual pipeline SIMD efficiency.

The 32 element BSB recall algorithm performs about 2240 floating operations for each recursive cycle; 2,176 for the actual algorithm and 64 for state vector conversions from and to integer fixed point. About 5 cycles are needed for convergence, yielding 11200 operations per 128 bytes of DMA data movement (no weight vector movement, and the state vector is actually 2 byte fixed point). Peak FLOP rate is (2176/4 + 64) 608 Quad Floating ticks/cycle. The peak DMA rate is (128/16) 8 DMA ticks. The peak compute to DMA ratio is therefore 608/8 = 76.

About 17 GFLOPs/Second (GFLOPS) were measured on the 128 element case. This corresponds to about 51,000 10 cycle recalls per second about $1/10^{th}$ the rate achieved using the FPGA. However, six of these can be run in parallel on a single PS3 node chip, bringing the throughput to about half of the FPGA case. The Cell chip is more than an order of magnitude less expensive than the FPGA chip, and the Cell chip is programmed in C, compared to VHDL needed for the FPGA. By these measures the Cell technology has significant cost advantages over the FPGA technology.

A trial was run using all 288 PS3 notes in AFRL/RI's Cell-BE cluster. The mark of 29.376 trillion FLOPS was reached.

About 11 GFLOPS were measured for the 32 element case. This corresponds to about 982,142 5 cycle recalls per second, about 1.5X faster than the FPGA doing the same work. However, since six of these can be performed in parallel in a PS3 node, the PS3 chip is potentially 9X faster than the FPGA.

Note that the 60 fold clock speed ratio (FPGA 100 MHz vs. Cell-BE SPE 6GHz) is a major factor in speed differences.

### 4.2.3 Confabulation acceleration

An effort was made to improve confabulation speed [51]. The original confabulation tools constructed during task 1 were used to collect usage statistics on sparse data. With these statistics the "lexicon" and "knowledge link" structures were recast from trees into hash tables, thus reducing some search times and use of indices. In general, a greater degree of data locality was achieved through hashing. A technique was developed to merge knowledge bases built of these structures.

Three strategies were explored for optimization of the sentence completion algorithm: software optimization, software analysis and hardware architecture augmentation. Our analysis shows there is potential to improve the three structure techniques using hashing strategies. The hashing strategies may improve data locality as well. A hash version of training was demonstrated in about 4 seconds, compared to the 45 seconds the tree structures used. The cogent confabulation algorithm is an ideal candidate for parallel processing. It also shows that although increasing the number of processors or the size of memories can increase the performance of training and recall, the relations between resource cost and performance associated with these variations are not always linear. The details of hardware configuration must be carefully considered to achieve good cost performance tradeoffs. We suggest that this work can be extended to more complex implementations of confabulation systems.

### 4.2.4 Hybrid minicolum: BSB + Neurons Acceleration

The BSB/Neuron hybrid model was implemented on a cluster of 12 dual quad 3 GHz Xeon platforms to the point where:

- All lateral and feed forward messaging was in place;
- The BSB part of the model was in place;
- Connection of simple cells to the BSB through a complex cell layer were in place;
- Lateral inputs were connected to the BSB;
- Lateral outputs were connected to their local neighborhoods of minicolumns;

**Table 12: Parvocellular simple cell receptive fields were in place**

| Orientation angle, in degrees | Number of receptive fields defined |
|---|---|
| 0 | 140 |
| 22 | 160 |
| 45 | 162 |
| 68 | 108 |
| 90 | 186 |
| 112 | 80 |
| 135 | 88 |
| 158 | 113 |

No Magnocellular components were yet implemented, and the Parvocellular receptive fields were rapidly prototyped. More work is needed there to improve the field placement and balance the number of fields in order to test efficacy properly. For example, it is probably best for all the orientation angles to be represented with the same number of receptive fields.

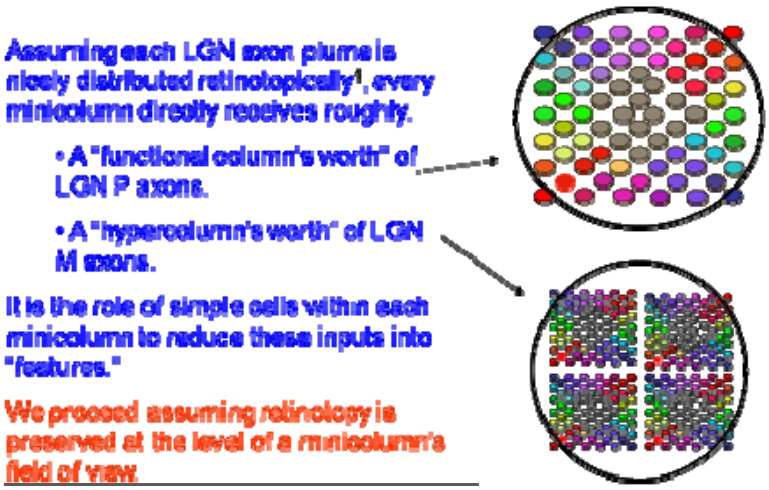



**Figure 42: Thalamic P and M channel ganglia spreads.**

In each orientation angle category the fields were evenly split between phase and antiphase directions: light to dark, dark to light. The receptive fields for Parvocellular ganglia (Figure 42) assume each minicolumn within a functional column assemble "see" all the Parvocellular ganglia entering the functional column area (0.4mm). (Magnocellular aperture is four functional columns worth (1.2mm) [41]. The field of view of a functional column is approximately 1:1 ratio of minicolumns to ganglia: about 64 "pixels."

Within a functional column, stacks of 8 minicolumns are arranged such that each column is an orientation column (responsive to a specific angle). The "randomly" arranged small P ganglia receptive fields covering a functional column area are assigned to the minicolumns in a manner where any minicolumn tends to "see" receptive fields that roughly line up. Each minicolumn receives 56 receptive fields; the pool of fields can have fields assigned to multiple minicolumns. Each receptive field has 8 synapses to ganglia.

The full scale model emulates stereo vision. Every other functional column row alternates between the ipsi-lateral and contra-lateral eye. The left hemispheric side of V1 sees the right field of view; the right hemispheric side of V1 sees the left field of view. There is a small overlap. This facility provides the means to study ocular disparity detection within minicolumns. The BSB state vector is set up as follows:

- Afferent field (the Parvo inputs): 8 elements.
- Disparity field: 4 elements.
- Lateral reception field: 8 elements.
- Extrastriate expectation data: 12 elements.

The full scale model is set up as a collection of subfields of a V1 region. Each subfield is a collection of 8192 minicolumns; 4096 associated with each eye and completely overlapping fields of view. Each subfield is implemented as a single LINUX process. Each subfield process receives its afferent data in a single frame; this is accomplished with a simple ocular chiasm model which accepts image frames from a model of each eye, then overlaps the frames as would a real ocular chiasm. It splits the two overlapped images into left and right halves with a small overlap. The chiasm process publishes each half as a separate message.

A separate process representing the *lateral geniculate nucleus* (*LGN*) subscribes to what the chiasm publishes. There are two LGNs in the thalamus; each subscribes to either a left or right field of view. The LGN tessellates a stereo field of view into stereo subfields whose tiles correspond to fields of view of the V1 subfields. At this point an LGN process publishes the individual tiles.

Each V1 subfield process subscribes to one LGN tile and V1 laterals. A V1 process publishes two types of information:

1. Feed forward percepts: destined for extrastriate cortex areas such as V2. Each of these message objects is "subfield size." It is up to the extrastriate models to subscribe to them in a retinotopic manner.

2. Lateral functional column consensus. Each functional column assembly of minicolumns computes a consensus of what its orientation columns see. Each consensus is an eight element vector corresponding to angle and direction perception. Each of these is destined for the functional column neighborhood of about 1500 functional columns, corresponding to a neighborhood of 14 surrounding subfields. Thus, each lateral publication of a subfield process is sent to 14 other subfields. This neighborhood extent corresponds to the 3.5 mm extent reported for V1 lateral connections [36].

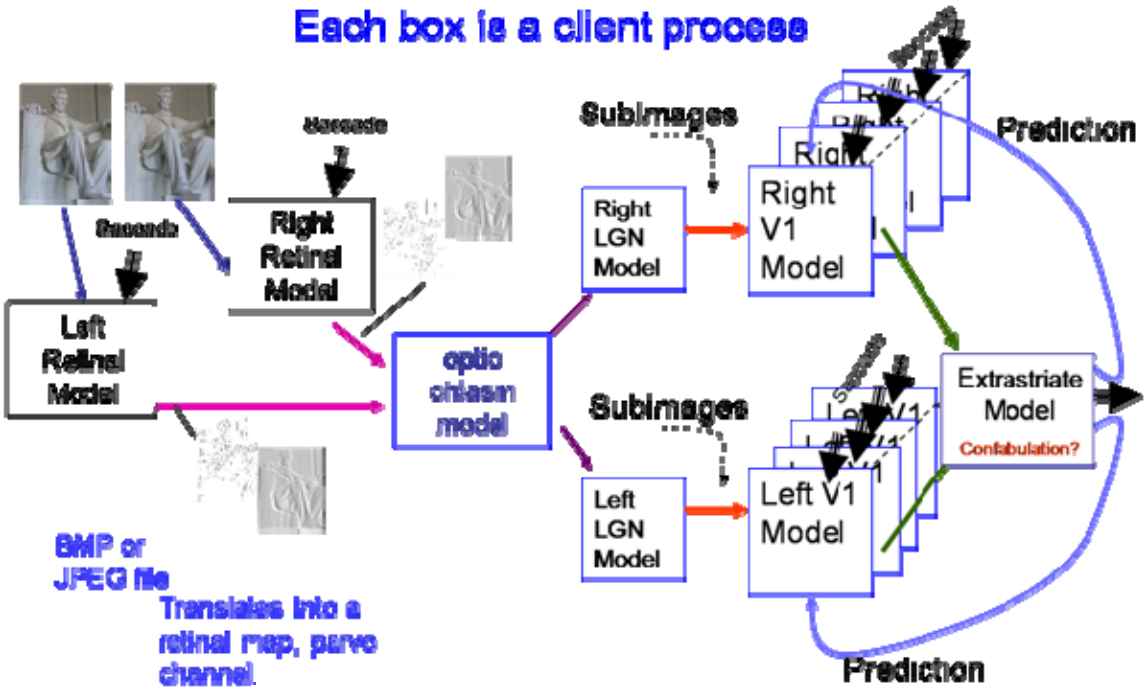



**Figure 43: Process infrastructure.**

The chiasm subscribes to images published by two sources: left and right eyes. The eye models are each a single LINUX process. At the moment, they are limited to still images, but there is no reason to maintain that limit other than project resource limitations. The eye models filter the incoming RGB images to provide P-channel (high spatial filter) and M-channel (low spatial filter) images. Each eye puts out about 0.8 million P-channel pixels, and about 1/16 of that for M-channel pixels. Each eye publishes a frame with both P and M channel information.

Large scale modeling will require significant visualization capability for model software debugging and model assessment. Some of this capability has been built to support the present modeling effort. In place are separate monitor processes capable of subscribing to V1 lateral and V1 feed forward percept messages. These processes symbolically display the message contents. In progress is a more general purpose visualization capability needed to support full scale assessment; the current capability is a component of the planned capability. Figure 43 illustrates the plan; the display on the right side of the figure, with blue and gray angle icons, is a subfield's worth of orientation percepts the current model computed for a subfield positioned over a section of the silhouette dog's hind leg. The illustrated angles roughly correspond to orientation perceptions and directions (This image off a debugging session). The fMRI coronal projection image in the same panel was is not actual test run data; it illustrates plans for improving the visualization capability

With visualization and the Pub/Sub server running on a dual quad platform with one retinal model, the chiasm process and one LGN process, a single subfield process was able to execute at about 2 frames per second. Real-time is probably 5 frames per second, corresponding to 5 saccades.

The focus then turned to porting the V1 subfield process to a SONY PlayStation® 3 Cell-BE platform. Twelve 24 node clusters of such Cell-BEs are available on subnets of the dual quad Xeon platforms. Each PS3 Cell-BE node has a Power PC core (PPE) and six satellite broad band engines (SPE). SPEs have small memories: 256K bytes, but can process floating point data rapidly (25.5 GFLOPs). Two high speed DMA channels (in, out) connect each SPE to a PPE. The PPE runs LINUX and has IP communication with the XEON head nodes and other PS3 nodes.

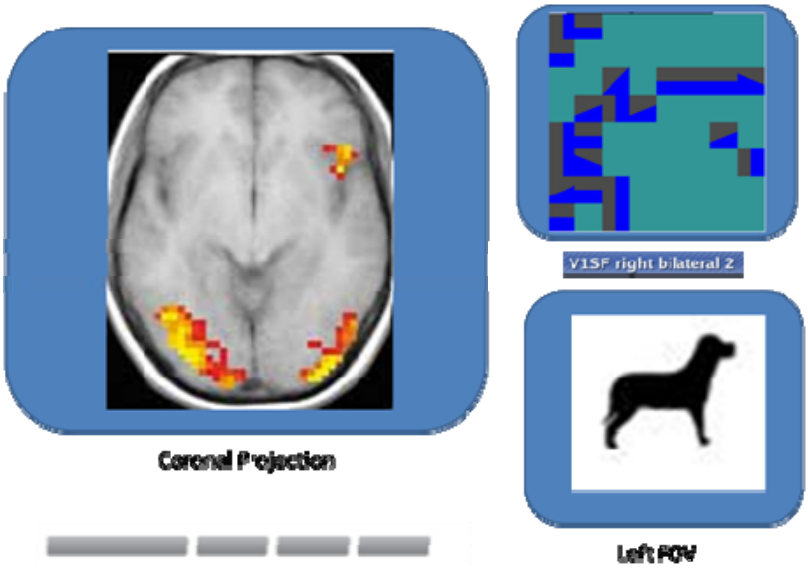



**Figure 44: Example visualization plans.**

The porting is being performed in three stages:

1. Movement of the subfield process onto the PPE element. This has been performed. The PPE is much slower than a dual quad Xeon and requires nearly 40 seconds for one frame. The "slowness" is likely due to memory thrashing (no attempt was made at optimization in this stage). A full scale 196 process V1 emulation was performed to verify messaging capability and performance. Parallel execution of the full scale model proceeded at about the same speed as a single subfield execution.

2. Porting of the BSB algorithm to the SPE side. The BSB algorithm had been shown to perform well on a Cell-BE SPE earlier in the study. All BSBs within a subfield share the same weight matrix, thereby sparing the overhead of uploading the weights for each minicolumn. This particular implementation has been completed and was measured to run at about 42% of peak. At this rate a single SPE should be able to provide real-time BSB performance for all 8192 BSBs within a subfield. The implementation accepts 16 bit fixed point BSE state vector values, converts them into floating point, performs a 5 recursion cycle BSB recall in floating point, and converts the new state vector back into fixed point.

3. Porting of the P-channel afferent processing to the SPE side. This effort has not yet commenced. It involves dealing with sparse vectors.

### 4.3 Task 3: Benchmarks

The cognitive benchmark leverage expected from the DARPA ACIP and BICA projects faded with the premature termination of those very large efforts. The alternative used on this effort has been to keep track of the testing methods used and those thought to be useful (if they were available) throughout the study, and compile a list of benchmark approaches which might be developed from what has been accomplished. These include:

A. A repository of freely available textual material; the text volumes within the repository can serve as data for repeatable experiments.
B. A repository of confabulation experiments, organized by type of experiment such as sentence completion
C. Orientation perception tests on standard images, both binocular and monocular. These tests can be aligned to some degree with clinical data taken on V1 response to high and low contrast grid lines oriented at various angles [26].
D. Ocular disparity detection within V1 can be benchmarked with clinical data comparison (Gonzalez & Perez, 1998).
E. Silhouette tracing – demonstrate V1 ability to detect orientation lines tracing a silhouette image at standard contrasts.
F. V1 ability to detect object edge boundaries in standardized natural scenes.

The standardization of test environment would facilitate sharing of benchmark capabilities. For example, visualization and image delivery mechanisms developed during this effort are useful infrastructure components. Levels of standardization should be managed: stimulation models (retina, chiasm, LGN) as well as the software implementing them are examples of two levels of management.

# 5. Conclusions

## 5.1 Confabulation

We believe that we captured the main elements of the confabulation training and recall algorithms working versions created during this project. Computationally, confabulation is characterized by vector x matrix dot product calculations carried out on operands stored with sparse array data storage methods. Confabulation may be useful to model cortex areas above the primary sensory fields, where multiple sensory mode integration starts.

We implemented, optimized, and evaluated the performance of the model in the context of two specific example application problems that we called sentence completion and intelligent on-line character recognition. Both problems required training by reading a corpus of electronic media such as books and newsfeeds. We found that training time for a single book is on the order of 1-10 seconds, and that knowledge base data storage requirements were basically linear with corpus byte count in early training, with evidence of size saturation with continued training.

Test scenario and metrics to judge the sensibility of confabulation performance were lacking, so we developed simple tests and metrics that graded the accuracy of sentence completion. Testing the confabulation model in a sentence completion application after training with a narrow training corpus, we found that the it can complete partial test sentences drawn from the training set with almost perfect accuracy (99%) with more than half of the words or characters deleted. Similarly, in the intelligent OCR application we also observed extremely high recall accuracies (~99%) for the case of completing test sentences with random word deletions of up to 60%, and (~99% accuracy) with random character deletions of up to ~20%. We did not observe degradation in recall accuracy with deeper training, at least up to about 40 books and newsfeed days. We conclude that the confabulation model operates as a reasonable content addressable memory on the scale of sentences.

We also determined that the model has good prospects for speedup and parallelization in on both hardware and multi-core platforms such as the IBM/Sony CBE. The confabulation training and recall models implemented in FPGA versions that achieved ~1,000x and ~3,000x speedup over sequential software, respectively. They also achieved ~ 5.8x speedups on one CBE chip, and there are good prospects for further speedup by parallelizing across a cluster of CBEs due to the regular arrangement of the lexicon and knowledge base elements in the models.

We believe that good solutions to the sentence completion problem could very well translate to other input modalities (i.e. audio and imagery), and to solutions in different application scenarios that are of interest to the military. For example, improved machine reading systems based on Optical Character Recognition, automatic textual annotation of surveillance motion imagery, prediction of sequences of events from video and conversational language translation systems [49].

## 5.2 Attractor Network Models

The attractor network idea has been examined from the point of view of neuromorphism (its place in a columnar reverse engineering model) and computational performance on several types of platforms. Computationally, it is feasible to scale up on the minicolumn or larger neuromorphic level. Neuromorphically, we agree these networks show promise. The types of attractors, which might be neurologically part of cognition: point, line or limit cycle, are still undetermined. It is not known to what extent the recurrent networks are randomly wired, and to what extent they follow a genetically predetermined architecture, which then is sharpened by learning.

If indeed the brain is constructed upon attractors as a basic building block, the science behind constructing a complex system of attractor networks is not yet developed. The one system concept that has been identified in this area is the "liquid state machine" [56]. The ideas associated with liquid state machines can be explored using attractors of any of these types within large scale assemblies; we now know we can emulate large scale systems of these networks.

## 5.3 Bayesian Network Models

The hierarchical Bayesian model, a tree, was examined from the point of view of computational performance on traditional platforms, and using FPGA augmentation. The model was greatly simplified by moving to a logarithmic representation that converted multiplies and divides into additions and subtractions. The model was then efficiently mapped to the FPGA resources, achieving over 90% utilization of the logic resources. A large scale cluster augmented with at least 2,000 FPGA devices would be needed for full neocortex emulation at real-time. By the end of this first 3 year investigation the CELL-BE technology had not been evaluated. This may be a good computational match. Bayesian methods may serve better to model higher level cognitive models (Cog-Psych Models) that are behaviorally based, and play a role in expectation in neuromorphic models development, as proposed for confabulation.

## 5.4 Hybrid BSB/Neuronal Models

A detailed full scale model of V1 is practical to host and run in real-time or near real-time on a cluster of about 200 Cell-BE processors. A cluster of 200 dual quad Xeon 3 GHz platforms will work as well. V2 modeling should be feasible using similar modeling.

The efficacies of the BSB/Neuron hybrid model have not yet been measured, but preliminary testing suggests silhouette orientation perception should work well and that some level of natural scene orientation efficacy will be possible even prior to including contrast control in the V1 model. Contrast control, disparity, and color/texture have yet to be modeled. More time is needed to explore the neurological data in order to produce neuromorphic models for these. Moreover, V1 interacts (principally) with V1 and V4v (form, color), and having at least simple models of these areas may be necessary to explore color and disparity perception. However, the infrastructure currently in place will facilitate that effort.

Computational underpinnings used for emulation have been vector matrix operations, floating point and fixed point operations, sparse data manipulation, and a very flexible messaging system.

The underpinnings being emulated include association (tightly reentrant network), integration (many to one spatial and temporal summation), non-linearity (neuron and confabulation thresholds, BSB limiting), loose recurrency (confabulation looping between "patches," lateral interactions and extrastriate loops within and with V1), and winner take all decisions. To some extent Liquid State Machines (LSM) are included because of the use of neuron assemblies "reading out" the BSB state vector into smaller vectors or scalars indicating features. LSM needs to be applied elsewhere, such as "reading out" attractors formed through the longer range, loose recurrencies in hierarchical circuits.

Neuromorphic computational architecture development is a new and accelerating field with significant promise. The Air Force needs to cultivate a community of expertise in order to exploit the emerging new technologies coming from it and related fields. Individual qualifications to contribute in this domain include familiarity in multiple disciplines such as: computer architecture/technology, parallel software development, dynamical systems, neuroscience, neurology, neuropsychology, and agent based expert systems.

# 6. Recommendations

A follow-on effort has been proposed to continue the development of V1 models, and add extrastriate modes. It also proposes to add another sensory mode: audition. This will improve the current columnar models, provide an increased scale and context for emulation studies, and add infrastructure for multimodal senses (vision and audition).

The testing infrastructure needs to be improved to support large scale parallel system debugging and validation, as well as efficacy testing.

Continued reliance on the Cell-BE clusters is recommended. This architecture seems to match well with the emulation computation needs, and it is a very cost efficient large scale system.

Confabulation is the method we investigated that is closest to high level cognition. Subjective measurements of efficacy are challenging to define. We recommend investigating Latent Semantic Analysis as a method of deriving objective metrics.

# 7. REFERENCES


[1] Achard S, Bullmore E. Efficiency and Cost of Economical Brain Functional Networks. PLoS Computational Biology Vol. 3, No. 2, e17 doi:10.1371/journal.pcbi.0030017

[2] Anderson, J.A. (1993). The BSB network. Pp. 77-103 in MH Hassoun (Ed.), Associative Neural Networks, New York, NY: Oxford University Press.

[3] Andreas V. M. Herz, et al. Modeling Single-Neuron Dynamics Computations: A Balance of Detail and and Abstraction DOI: 10.1126/science.1127240 Science 314, 80 (2006);

[4] Arbib, M. A., & Grethe, J. S. (Eds.), (2001). Computing the brain: A guide to neuroinformatics. San Diego: Academic Press.

[5] Bassett, Danielle S. Meyer-Lindenberg, Andreas. Achard, Sophie. Duke, Thomas and Bullmore, Edward. "Adaptive reconfiguration of fractal small-world human brain functional networks." PNAS | December 19, 2006 | vol. 103 | no. 51 | 19518-19523

[6] Buzás P, Eysel UT, Adorján P, Kisvárday ZF (2001) Axonal topography of cortical basket cells in relation to orientation, direction, and ocular dominance maps. J Comp Neurol. 2001 Aug 27;437(3):259-85

[7] Crair, M. C. ; Gillespie, D. C. and Stryker, M. P. The role of visual experience in the development of columns in cat visual cortex, Science (1998), 279:566–570.

[8] Dean, T. Hierarchical Expectation Refinement for Learning Generative Perception Models. Tech. rep., Brown University, Providence, Rhode Island, Aug 2005.

[9] DeFelipe, J., Hendry, S.H.C., Hashikawa, T., Molinari, M. and Jones, E.G., 1990. , A microcolumnar structure of monkey cerebral cortex revealed by immunocytochemical studies of double bouquet cell axons. Neuroscience 37, pp. 655–673 Abstract | PDF (5472 K) | View Record in Scopus | Cited By in Scopus (71)

[10] Dickison, Daniel, Taatgen, Niels. ACT-R Models of Cognitive Control in the Abstract Decision Making Task. Proceedings of ICCM - 2007- Eighth International Conference on Cognitive Modeling. 79 - 84. Oxford, UK: Taylor & Francis/Psychology Press

[11] Douglas, Rodney J. Martin, Kevan A.C. Recurrent neuronal circuits in the neocortex. Current Biology Vol 17 No 13. R496, 2007.

[12] Engel, S. A., Glover, G. H. and Wandell B. A., "Retinotopic organization in human visual cortex and the spatial precision of functional MRI," Cerebral Cortex, vol. 7, pp. 181–192, 1997.

[13] Frans´en E, Lansner A. 1998. A model of cortical associative memory based on a horizontal network of connected columns. Network: Comput Neural Systems 9:235–264.

[14] Freeman, W.J. 2006. Origin, structure, and role of background EEG activity. Part 4: Neural frame simulation. Clinical Neurophysiology 117(March):572-589. Abstract available at http://dx.doi.org/10.1016/j.clinph.2005.10.025.

[15] George, Dileep and Hawkins, Jeff. A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. In Proceedings of the International Joint Conference on Neural Networks. IEEE, 2005.

[16] Gonzalez, Francisco & Perez, Rogelio. Modulation of cell responses to horizontal disparities by ocular vergence in the visual cortex of the awake macaca mulatta monkey. Neuroscience Letters, Volume 245, Issue 2, 3 April 1998, Pages 101-104

[17] Hasson, Uri; Yang, Eunice; Vallines, Ignacio; Heeger, David J. and Rubin, Nava. A Hierarchy of Temporal Receptive Windows in Human Cortex. The Journal of Neuroscience, March 5, 2008, 28(10):2539-2550; doi:10.1523/JNEUROSCI.5487-07.2008

[18] Hawkins, Jeff and Blakeslee, Sandra, "On Intelligence," Times Books, Henry Holt and Company, New York, NY 10011, Sept 2004.

[19] Hecht-Nielsen R., Cogent confabulation, Neural Networks, 18 (2005), pp. 111–115.

[20] Hecht-Nielsen R., Mechanism of Cognition. In: Bar-Cohen, Y. [Ed.] Biomimetics: Biologically Inspired Technologies, CRC Press, Boca Raton, FL (2006).

[21] Hodgkin, A., and Huxley, A. (1952): A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117:500–544.

[22] Horton, J.C. and Hocking, D.R. An adult-like pattern of ocular dominance columns in striate cortex of newborn monkeys prior to visual experience, J. Neurosci. 16 (1996), pp. 1791–1807

[23] Hubel DH, Wiesel TN. Receptive fields of single neurons in the cat's striate cortex. J Physiol. 1959 Oct;148(3):574–591.

[24] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol. 1962 Jan;160(1):106–154.2.

[25] Hubel DH, Wiesel TN. Shape and arrangement of columns in cat's striate cortex. J Physiol. 1963 Mar;165(3):559–568.2.

[26] Hubel DH, Wiesel TN.. Receptive Fields and FunctionalL Architecture in two Nonstriate Visual Areas (18 AND 19) of the Cat. J Neurophysiol. 1965 Mar;28:229–289.

[27] Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex. J Physiol. 1968 Mar;195(1):215–243.

[28] Hubel DH, Wiesel TN. Sequence regularity and geometry of orientation columns in the monkey striate cortex. J Comp Neurol. 1974 Dec 1;158(3):267–293.

[29] Hubel DH, Wiesel TN, Stryker MP. Anatomical demonstration of orientation columns in macaque monkey. J Comp Neurol. 1978 Feb 1;177(3):361–380.

[30] Izhikevich EM , "Simple Model of Spiking Neurons", IEEE Transactions on Neural Networks 14:1569-2003.

[31] Izhikevich, Eugene M. Polychronization: Computation With Spikes. Neural Computation (2006) 18:245-282

[32] Izhikevich, Eugene M. Which Model to Use for Cortical Spiking Neurons? IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 15, NO. 5, SEPTEMBER 2004

[33] Izhikevich, Eugene M. Simple Model of Spiking Neurons. IEEE Transactions on Neural Networks, Vol. 14, No. 6, November 2004

[34] Johansson, Lansner Imposing Biological Constraints onto an Abstract Neocortical Attractor Network Model (Neural Computation. 2007;19:1871-1896.)

[35] Kayser, Priebe, and Miller. Contrast-Dependent Nonlinearities Arise Locally in a Model of Contrast-Invariant Orientation Tuning. Journal of Neurophysiology. 2001.

[36] Kisvarday, et al. One axon-multiple functions: Specificity of lateral inhibitory connections by large basket cells. Journal of Neurocytology 31, 255–264 (2002)

[37] Kitano, K., Fukai, T. A multiple synfire-chain model for the predictive synchrony in the motor-related cortical areas. Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on. Publication Date: 18-22 Nov. 2002, Volume: 4, On page(s): 1634- 1638 vol.4

[38] Lafontant ,S. and Taha, T. M. "Feasibility of Hardware Acceleration of a Neocortex Model," The International Conference on Engineering of Reconfigurable Systems and Algorithms, June 2007.

[39] Lee, Tai Sing and Mumford, David. Hierarchical Bayesian inference in the visual cortex. Journal of the Optical Society of America, 2(7):1434–1448, July 2003.

[40] Lu, Haidong D. and Roe, Anna W.. Functional Organization of Color Domains in V1 and V2 of Macaque Monkey Revealed by Optical Imaging. Cerebral Cortex Advance Access originally published online on June 18, 2007. Cerebral Cortex 2008 18(3):516-533; doi:10.1093/cercor/bhm081

[41] Lund, Jennifer S.; Angelucci, Alessandra & Bressloff, Paul C. Anatomical Substrates for Functional Columns in Macaque Monkey Primary Visual Cortex. Cerebral Cortex, Vol. 13, No. 1, 15-24, January 2003.

[42] Maass, W., Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation, 14(11):2531-2560, 2002.

[43] Maass, W., Natschlager, T., and H. Markram. Computational models for generic cortical microcircuits. In J. Feng, editor, Computational Neuroscience: A Comprehensive Approach. CRC-Press, 2002.

[44] Maass, W. Legenstein R. A., and Markram H.. A new approach towards vision suggested by biologically realistic neural microcircuit models. In Proc. of the 2nd Workshop on Biologically Motivated Computer Vision, Lecture Notes in Computer Science. Springer, Nov. 2002.

[45] Martínez, José F. Research Gap Analysis from the "Workshop on Research Directions in Architectures and Systems for Cognitive Processing.", Cornell University, July 2005, Ithaca, NY

[46] Mountcastle V.B. The columnar organization of the neocortex. Brain. 1997;120:701–722.

[47] http://bluebrain.epfl.ch/

[48] Pearl, Judea. "Probabilistic Reasoning in Intelligent Systems," Morgan Kaufman Publishers, San Francisco, California, 1988.

[49] Pinto, Hugo; Wilks,Yorick & Catizone, Roberta. "The senior companion multiagent dialogue system", In Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008).

[50] Pournara, Iosifina, Bouganis, Christos-S, and Constantinides, George A., "Fpga-Accelerated Bayesian Learning For Reconstruction of Gene Regulatory Networks," in International Conference on Field Programmable Logic and Applications, 2005.

[51] Qinru Qiu, Daniel Burns, Michael Moore, Richard Linderman, Thomas Renz, Qing Wu. Accelerating Cogent Confabulation: an Exploration in the Architecture Design Space. 2008 Intl Joint Conference on Neural Networks, (IJCNN) at the 2008 IEEE World Congress on Computational Intelligence (WCCI).

[52] Rehder, B., Schreiner, M. E., Wolfe, M. B., Laham, D., Landauer, T. K., & Kintsch, W. (1998). Using Latent Semantic Analysis to assess knowledge: Some technical considerations. Discourse Processes, 25, 337-354. See also http://lsa.colorado.edu/

[53] Robert, P.D. and Bell, C.C: Spike timing dependent synaptic plasticity in biological systems, Biol. Cybern. 87 (2002), pp. 392–403

[54] Serre, Thomas. "Learning a Dictionary of Shape-Components in Visual Cortex:Comparison with Neurons, Humans and Machines. MIT-CSAIL-TR-2006-028, CBCL-260, April 25, 2006

[55] Shepherd, Mirsky, Healy, Singer, Skoufos, Hines, Nadkarni, Miller. The Human Brain Project: neuroinformatics tools for integrating, searching and modeling multidisciplinary neuroscience data. Trends in Neurosciences, Volumn 21, Issue 11, 1 Nov 1998, Pages 460-468.

[56] Yamazaki, Tadashi; Tanaka, Shigeru. The cerebellum as a liquid state machine. Neural Networks. Volume 20, Issue 3, April 2007, Pages 290-297

[57] Wikemedia. http://commons.wikimedia.org/wiki/Image:Brain-anatomy.jpg

[58] Willshaw D.J., Buneman O.P., Longuet-Higgins H.C. (1969) Non-holographic associative memory. Nature 222:960-962.

[59] Wu, Qing; Mukre, Prakash; Linderman, Richard; Renz, Tom; Burns, Daniel; Moore, Michael; Qiu,Qinru; Performance Optimization for Pattern Recognition Using Associative Neural Memory. IEEE International Conference on Multimedia and Expo, 2008. On pages: 1-4. Publication Date: June 23 2008-April 26 2008.

[60] Yabuta N H; Sawatari A; Callaway E M, "Two functional channels from primary visual cortex to dorsal visual cortical areas", Science (New York, N.Y.) 2001;292(5515):297-300.

[61] Yao, Xingzhong; Jin, Lianghai and Hu, Hanping. Pinwheel patterns give rise to the direction selectivity of complex cells in the primary visual cortex. Brain Research Volume 1170, 19 September 2007, Pages 140-146

[62] Xilinx. http://www.xilinx.com/products/boards/ml310/current/

[63] Bear, Connors, Paradiso. Neuroscience, Exploring the Brain. Second edition. Lippincott Williams & Wilkins. Baltimore. 2001

[64] Qing Wu, Qinru Qiu, Richard Linderman, Daniel Burns, Michael Moore, Dennis Fitzgerald. "Architectural Design and Complexity Analysis of Large-Scale Cortical Simulation on a Hybrid Computing Platform." IEEE Computational Intelligence for Security and defense Applications (CISDA), 2007.

[65] Richard Linderman. Qing Wu, Qinru Qiu, "FPGA and Cell Processor Performance Optimization for Brain-State-in-a Box (BSB) cognitive Computing", 2007 ARCS Symposium on Multicore and New Processing Technologies, Aug 2007..

# 8.0 Appendix 1

Publications/ Reports

**2007:**

- Sebastien Lafontant and Tarek M. Taha, Clemson University. "Feasibility of Hardware Acceleration of a Neocortex Model." World Congress in Computer Science, Computer Engineering, & Applied Computing. Las Vegas, Nevada, USA (June 25-28, 2007)
- Qing Wu, Qinru Qiu, Richard Linderman, Daniel Burns, Michael Moore, Dennis Fitzgerald. "Architectural Design and Complexity Analysis of Large-Scale Cortical Simulation on a Hybrid Computing Platform." IEEE Computational Intelligence for Security and defense Applications (CISDA), 2007.
- Richard Linderman. Qing Wu, Qinru Qiu, "FPGA and Cell Processor Performance Optimization for Brain-State-in-a Box (BSB) cognitive Computing", 2007 ARCS Symposium on Multicore and New Processing Technologies, Aug 2007.

**2008:**

- Wu, Qing; Mukre, Prakash; Linderman, Richard; Renz, Tom; Burns, Daniel; Moore, Michael; Qiu,Qinru; Performance Optimization for Pattern Recognition Using Associative Neural Memory. IEEE International Conference on Multimedia and Expo, 2008. On pages: 1-4. Publication Date: June 23 2008-April 26 2008.
- Qinru Qiu, Daniel Burns, Michael Moore, Richard Linderman, Thomas Renz, Qing Wu, "Accelerating Cogent Confabulation: an Exploration of the Design Space", 2008 Intl Joint Conference on Neural Networks, (IJCNN) at the 2008 IEEE World Congress on Computational Intelligence (WCCI), June, 2008..
- Vutsinas C. N., Rice, K. L., and Taha T. M., "A Streaming Architecture for Cognitive Computing on the Cray XD1," Reconfigurable Architectures Workshop (RAW), Reno, NV, (April 2008).
- Rice, K, Vutsinas, C., and Taha, T. M., "A Scaling Analysis of a Neocortex Model Implementation on the Cray XD1," Journal of Supercomputing.
- Vutsinas, C., Rice, K, and Taha, T. M., "A Context Switching Streaming Memory Architecture to Accelerate a Neocortex Model," Journal of System Architecture.